uc3m | Universidad Carlos III de Madrid

Bachelor Thesis

Degree in Aerospace Engineering

Trajectory predictability analysis under convective weather conditions

Juan Rodríguez Carrillo

TUTORS

Daniel González Arribas

Javier García-Heras Carretero

2017 - 2018

Abstract

Weather prediction uncertainties are a limitation and a challenge for Trajectory-Based Operations. Understanding the effect of these uncertainties is essential in order to improve the reliability and predictability of the Air Traffic Management system. In this project convective regions are considered as the unique source of uncertainty. The scope of the project is to relate the effect of these regions to variations in flight trajectories with respect to the planned ones.

The project can be divided into three parts: the first one requires gathering flight information and structuring it in an organized way. The second one consists on obtaining meteorological information and associating a factor of risk to each flight. This factor is a measurement of the exposure to potential convective storms. Finally, this factor of risk is related to relevant aspects of the flight, such as delays or changes in the route.

The results obtained in this project could contribute to an enhancement of Trajectory-Based Operations, allowing a better understanding of the effect of these uncertainties in aircraft trajectories, improving therefore, the predictability of the Air Traffic Management system.

Key words: Aircraft navigation; Air Traffic Management; Forecast uncertainty; Convective regions;

Acknowledgments

First of all, I would like to thank the two persons who have made this project possible: my two tutors, Daniel González and Javier García-Heras. I want to express my gratitude for all your help. Thanks for finding time in your schedules to solve the different issues we encountered and for giving me your valuable advice.

Thanks also to all my professors in Universidad Carlos III de Madrid for being an inspiration in their classes, not only teaching the different topics of the subjects, but also true engineering thinking. In addition, thanks to my physics teacher Guillermo Martínez from high school who taught me to work hard for his scary exams.

Having almost finished the degree I realize how many people have been part of this important cycle. Thanks to Cristóbal, Mercedes, Enrique, Elena and Alberto for having made these years less hard. Without you it wouldn't have been the same. Thanks to the delegates Daniel and Manuel for being so patient with my questions about absolutely everything. Also thanks to Daniel Sánchez-Biezma for inspiring me with his passion for aircraft.

Furthermore, thanks to the Residence Hall Fernando Abril Martorell for having been a home for three years. There, I met people with great hearts and lots of dreams. Thanks to Carmen, Noemi, Rodri, Dani and Cris. Also thanks to my roommate Samuel for being like a little brother.

Thanks to my flatmates, Iñigo and Diego for making this last year of university special. We have lived together many experiences, the good and the bad. Thanks for the laughs, for teaching me how to cook and for all your support and advice. Also I want to thank Patricia for being like the fourth member of this apartment.

Moreover, I wouldn't be the person I am today without the people I grew up with. Thanks to Antonio, Juan Francisco and Óscar for being always there. Thanks to Mario, Samuel, Jorge and Gonzalo for all the moments lived. And thanks to Alba and Elena for always believing in me.

Finally, I want to thank the people that have never stopped supporting and encouraging me: my family. Thanks María for taking care of me and cheering me up. Thanks dad for teaching me the values which define me today. And thanks mom for your passion and care, and for teaching me to never give up. There are no words to describe the pride and gratitude I feel for my family. You are the best family I could have asked for.

Contents

List of Figures

List of Tables

1 Introduction

Air transport has become essential for the current globalized world due to its ability to cover wide distances all over the planet, connecting people, industries and resources from different points of the globe. It is the fastest way to travel long distances, and also the safest. However, the user perception of the air transport is negative due to the long queues at the security controls and boarding gates, congested and wrongly designed airports compromising their capacity and the delays. There are many situations that can cause a delay on a flight, such as congestion of the airspace, technical failure or unfavorable weather conditions.

This report focuses on the delays and changes in the trajectories caused by convective weather conditions. Convection is defined as the transport of some property (usually heat) by fluid movement. In meteorology, convection is usually referred to heat transport by the vertical component of the flow [1]. Hazardous weather events (dangerous wind gusts, thunderstorms, heavy rainfall, etc) can be produced by deep moist convection.

Turbulence, one of the consequences of convection, costs airlines hundreds of millions of dollars in terms of unplanned maintenance, damages and delays [2]. It is remarkable that when maintenance is required, the cost for the airline is not just the cost of the repair but the lost revenue while the aircraft is not in service. Turbulence has different degrees of severity, implying changes in altitude, attitude and speed. In the most extreme case, turbulence causes structural damage and the pilot may lose the control of the aircraft.

Other damaging effects of convection are thunderstorms, hail and icing. Lightning strikes may harm the electrical system, requiring maintenance. The strong gusts in a thunderstorm keep hail suspended in the air, prone to impact the aircraft fuselage. Hail damages the radome and windshields. Lastly, ice formation on the surface of the aircraft modifies the aerodynamic shape, increasing the drag and decreasing the lift. Some effects of ice formation are wing stall, tail stall and loss of roll control. In addition, sensors such as static ports and pitot tubes may be blocked due to ice formation, providing erroneous indications.

The scope of this project is to perform a statistical analysis on the predictability (2 - 3 hours before the departure) of the delays caused by these adverse weather conditions. For each flight evaluated, a factor of risk of exposure to convective weather will be computed based on weather forecasts.

The intended trajectories filed in the flight plans will be considered, estimating the exposition to convection in the route. The real trajectories will vary in order to avoid the regions where convective storms are produced. The higher the exposure to convective regions, the higher the probability of changing the route.

Finally, the statistical study will provide an idea of the role of the weather forecasts on predictability of trajectories and delays. The aim will be to identify the

cause-effect relationship between the factor of risk and different flight parameters such as the difference between the estimated and actual take-off time, length of the route and final delay.

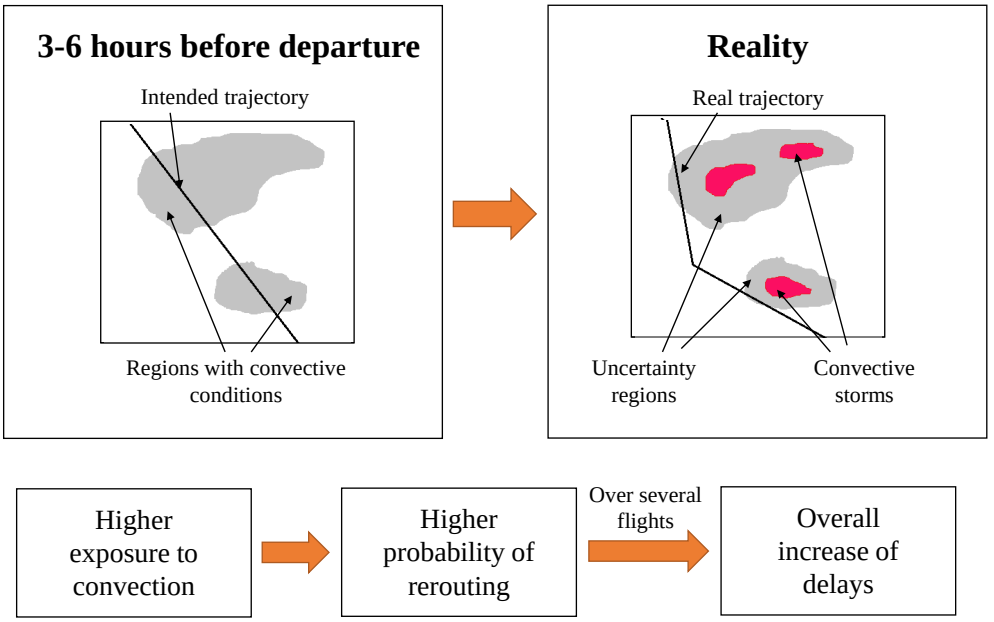


Figure 1: Concept of the project

2 State of the Art

This project tries to provide results which could contribute to an improvement on Trajectory-Based Operations (TBO). TBO is the exchange, maintenance and use of consistent flight information for collaborative decision-making on the flight [3]. TBO plays a dominant role in Air Traffic Management (ATM) strategies and concepts.

One of the most demanding tasks of TBO is the presence of uncertainties in the model, complicating ATM labour. It is paramount to understand the effect of this uncertainties to improve the predictability of the ATM system and, as a consequence, enhance factors such as capacity and efficiency.

There have been studies analyzing different sources of uncertainty in the models. For instance, [4] proposes a methodology to asses the impact of aircraft performance uncertainty on trajectory prediction accuracy. It enables the definition of bounds in performance uncertainty and trajectory prediction error. Another relevant study is given in [5], which analyzes the initial mass uncertainty. In addition, the uncertainty of the aircraft's planned future behaviour, or aircraft intent is studied in [6].

For this project, the uncertainties of interest are meteorological, one of the most relevant sources affecting ATM systems. The currently developing project SESAR ER TBO-Met ([7]) analyzes and quantifies the effects of weather prediction uncertainty in TBO.

In the area of weather forecasts, Numerical Weather Prediction (NWP) centers developed Ensemble Prediction Systems (EPS) as a way to provide probabilistic meteorological forecasts. In EPS, the NWP model is run several times varying some characteristics in order to obtain a set of forecasts. In that way, the ensemble can be probabilistic. A wide explanation of the status of NWP and the relevance of EPS in a meteorological context is given in [8].

In the last years, EPS have started to be used by ATM researchers in studies of predictability of flight plans and sensitivity to meteorological uncertainties. A promising approach to probabilistic trajectory predictor systems is IMET, a SESAR WP-E project. Its preliminary results were published in [9]. It showed how this approach can be used in pre-tactical level to improve collaborative decision-making.

The SESAR ER TBO-Met project studies two sources of meteorological uncertainty: wind and convective regions. UC3M has studied both sources of uncertainty within the framework of SESAR's TBO-Met project. Wind uncertainty was first analyzed in [10] and [11], developing a robust route optimization in aircraft trajectory planning under wind uncertainty. Afterwards, in [12], this methodology was extended in order to allow for convection risk.

In this project the wind uncertainties are not considered. The contribution of the project consists on determining the predictability due to convection of the trajectories at pre-tactical level combining real aircraft trajectories and EPS.

3 Methodology

The whole project can be divided into three main parts. The first task of the project is to obtain the desired data for analysis of the flights in the sample, and to store that data in a convenient way to be able to work with it. The second part consists on computing the risk of convection for each trajectory, using weather forecasts for specific dates and locations. Finally, once the parameter of risk is obtained, it is correlated with the relevant data of the flight (route distance, duration of flight, delay, etc.).

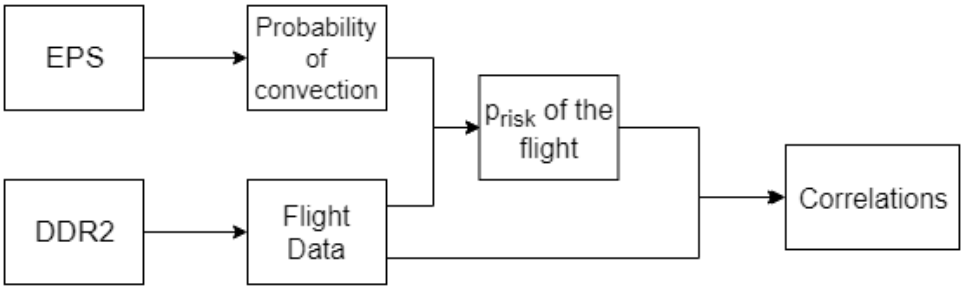


Figure 2: Scheme of the procedure followed

The programming language used to handle the data and do the necessary computations is *Python*. It is an open source language easy to manage and useful to work with large amounts of data of different types thanks to the libraries *NumPy* and *Pandas*. In addition, the library *Datetime* is quite functional when working with the date and time of a flight.

3.1 Flight data acquisition

3.1.1 Data retrieving

In order to perform the proposed analysis, data of a large amount of flights is needed. As mentioned before, the year 2017 is taken as the sample to study. A whole year is selected in order to try to cover all the possible weather scenarios.

The required data is provided by Eurocontrol [3], which is is an intergovernmental European organization whose main purpose is to consolidate the ATM in the European region. It offers flight data accessible for research or academic purposes. It is possible to access this information through its extranet *OneSky*, where an account is required. The *OneSky* account allows to download different data files related with ATM through the tool *DDR2 - Demand Data Repository*.

The files of interest for the project are found on the section *Historical Traffic*, where flight data from 2011 to current date can be downloaded in different formats. ALL_FT+ format is the most suitable for this project since it contains the most complete set of data. Figure (3) shows the DDR2 interface, where it is possible to download a file in the desired format for a specific day.

ALL_FT+ files contain flights focused on the European region. However, some flights may depart from or arrive at airports located in different regions.

Initially, the aim was to collect data of the whole year 2017. However, downloads in the DDR2 were limited to five files per week. This issue reduced the sample and some days were dismissed.

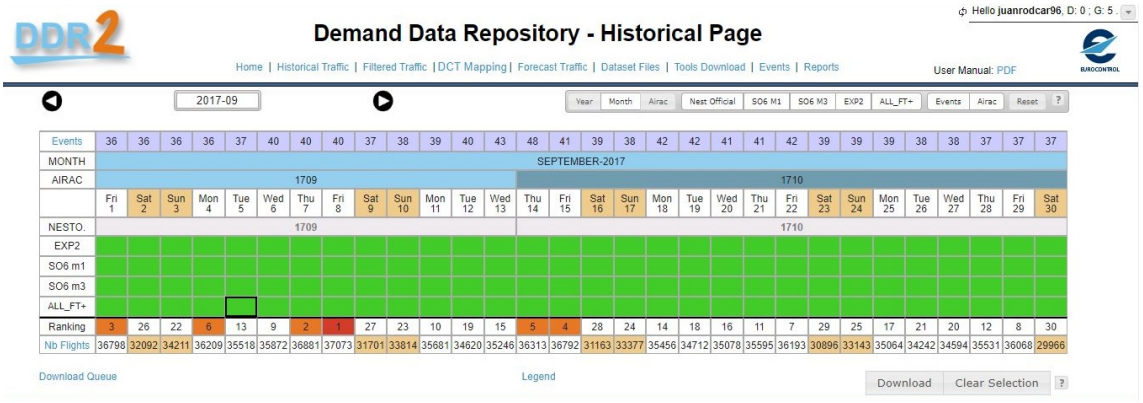


Figure 3: Eurocontrol: Demand Data Repository

ALL_FT+ files can be read using Python as text files. The format of these files is depicted in the DDR2 user manual [13]. Each line on the file contains 172 fields of information of one flight, where each field is separated by a semicolon (";"). The relevant fields used in this project are the following:

- Departure aerodrome (ICAO airport code).
- Arrival aerodrome (ICAO airport code).
- Aircraft operator (ICAO airline designator).
- Aircraft type (ICAO aircraft designator).
- Estimated Off-Block Time (EOBT). The estimated time at which the aircraft initiates taxiing manoeuvres before departure, determined in the flight plan. Format: [YYYYMMDDhhmmss].
- Actual Off-Block Time (AOBT). The real time at which the aircraft initiates movement. Format: [YYYYMMDDhhmmss].
- Filed Tactical Flight Model (FTFM): Point Profile. List of waypoints of the trajectory determined in the flight plan.

- Current Tactical Flight Model (CTFM): Point Profile. List of waypoints of the actual trajectory of the flight.

The fields of the point profiles contain the information of several waypoints. A waypoint is a specified geographical location used to define an area navigation route. For each waypoint, some subfields are divided by a colon (':'). The subfields of interest are:

- Time. Date and time at which the aircraft reaches the waypoint (or intends to reach it). Format: [YYYYMMDDhhmmss].
- Name of the waypoint (if applicable).
- Flight level.
- Coordinates.

3.1.2 Computation of additional parameters

Once the data is extracted from the ALL_FT+ files, other relevant variables can be obtained. The time subfield of the first and last waypoint in a point profile are used to obtain the take-off and landing times respectively. From FTFM, the Estimated Take-Off Time (ETOT) and Estimated Landing Time (ELDT) are acquired, whereas from the real trajectory, the actual times (ATOT and ALDT) are retrieved.

In addition, the distance covered by the trajectory is also estimated. This is done by adding the orthodromic distance between two consecutive waypoints. This distance can be computed using the Haversine formula (equations 1-3), applied to calculate distances in spherical trigonometry [14]. For two points with latitudes ϕ_1, ϕ_2 and longitudes λ_1, λ_2, given the radius of Earth ($R = 6371\ km$), the shortest distance joining the two points is given by:

$$a = \sin^2\left(\frac{\phi_2 - \phi_1}{2}\right) + \cos(\phi_1)\cos(\phi_2)\sin^2\left(\frac{\lambda_2 - \lambda_1}{2}\right) \tag{1}$$

$$c = 2\arctan\left(\frac{\sqrt{a}}{\sqrt{1-a}}\right) \tag{2}$$

$$d = R \cdot c \tag{3}$$

Furthermore, in order to analyze the impact of the studied parameters in different types of aircraft, each aircraft is classified according to the ICAO wake turbulence category (WTC), based on the maximum take-off mass of the aircraft:

- Light (L): Less than 17000 kg
- Medium (M): Less than 136000 kg and more than 17000 kg

- Heavy (H): More than 136000 kg

In that way, it can be possible to determine if the mass of the aircraft plays an important role in the trajectory under convective weather. The WTC of an aircraft is provided by ICAO [15].

Although in practice WTC is employed to provide appropriate separation between aircraft in take-off and landing, in this project it is used to differentiate among diverse types of routes. Heavy aircraft will cover longer distances at higher altitudes than light aircraft. In this way, the influence of the factor of risk in the variables studied can be established for each type of aircraft.

3.1.3 Structuring the data

Figure (4) shows the scheme of the code used to extract the data from the ALL_FT+ files. It is composed of two loops, one for the date, in charge of loading a file for each date downloaded, and an inner loop which reads the flights, storing its information.

In order to manage the information of each flight, a class named *Flight* is defined, containing the following attributes:

- Departure aerodrome
- Arrival aerodrome
- Aircraft type
- Aircraft WTC
- EOBT
- AOBT
- ETOT
- ATOT
- ELDT
- ALDT
- Estimated route distance
- Actual route distance
- Factor of risk (To be computed afterwards)
- Filed waypoints
- Current waypoints

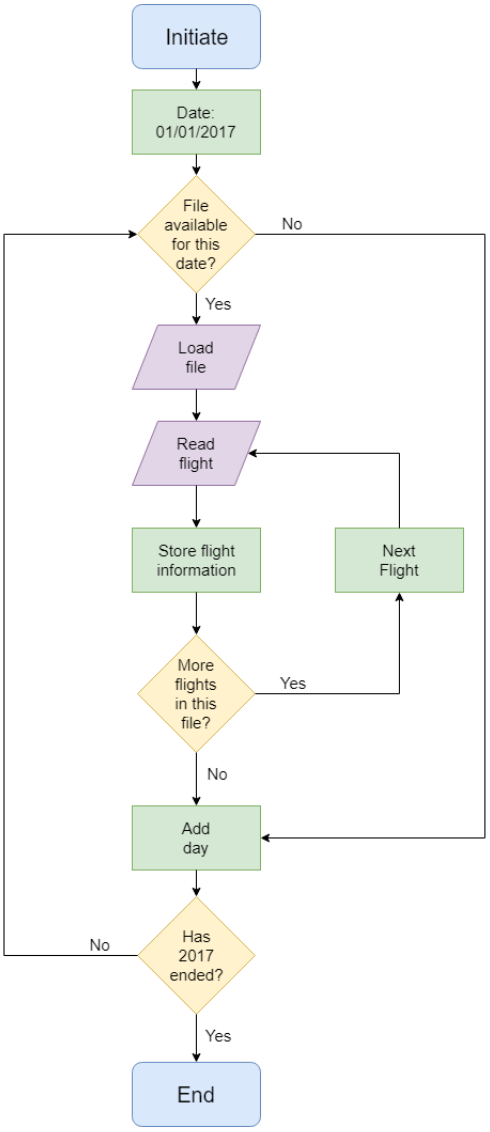


Figure 4: Data acquisition flowchart

Each *Flight* instance is stored in a list of flights. Also, the attributes referring to point profiles (waypoints) are lists of instances of another defined class called *Wpt*. The class *Wpt* contains the following attributes:

- Time
- Longitude
- Latitude
- Name
- Flight level

There are some special cases of aerodromes and aircraft types not registered in ICAO.

For these special cases, the designator assigned is ZZZZ [16]. In the case of aircraft type, aircraft having this designator is treated as category M (Medium). Flights departing from or arriving at aerodromes with this designator have an additional complexity: the waypoint corresponding to the not registered aerodrome does not have other information than the name in the ALL_FT+ files. Thus, the coordinates and time of these waypoints cannot be used. The same issue is found in flights departing from an airport with the designator AFIL, which is used when the flight plan is filed in the air.

In the case of departing from a ZZZZ or AFIL airport, the take-off time is considered that of the second waypoint in the point profile trajectory. On the other hand, the landing time of flights arriving at a ZZZZ airport is considered to be the time of the penultimate waypoint.

At this point, the data acquisition phase is finished and the next task is to obtain the parameter of risk of suffering convective weather.

3.2 Computation of the factor of risk

In this section it is explained how the factor of risk of exposure to convective weather is computed for each trajectory. From this point on, this factor of risk will be referred as p_{risk}.

The first task of this section is to obtain weather forecasts. The computation of the p_{risk} cannot be achieved using deterministic forecasts. Thus, probabilistic predictions are required. They can be retrieved from the TIGGE dataset, ECMWF EPS forecast [17]. Weather forecasts should cover the whole globe. Several files are required since there are two factors to be considered:

- The date and time at which the forecast is published.
- The time for which the forecast is made.

For example, a forecast may be published the first of January at 00:00 but the weather predicted is for six hours later.

The criteria to select the files for the forecast is the following:

- For each day studied, forecasts published at 00:00 and 12:00.
- Forecasts from 0 to 24 hours with a time step of 6 hours.

The time of publication of the forecast is selected in terms of the EOBT. The flight plan is filed three hours before the departure of the flight, that is, three hours before EOBT. This is considered the initial time t_0 of the flight. Thus, the forecasts used in a flight are the most recent forecasts published at t_0. The time of publication of the forecasts used in a flight is taken as the reference time.

In that way, a flight departing the second of January at 01:00, has as initial time t_0, 22:00 of the first of January. The most recent available forecasts at t_0 are published on the first of January at 12:00.

Once the reference time is determined, each waypoint is evaluated in order to select its suitable forecast. The difference (in hours) between the time of the waypoint and the reference time is approximated to the closest multiple of six. Mathematically, the expression which approximates an arbitrary number x to the closest multiple of an integer number N is:

$$a = N \cdot floor\left(\frac{x}{N} + \frac{1}{2}\right) \tag{4}$$

where a is the value approximated and *floor* is a function which converts a float to an integer by truncation.

If the aircraft of the previous example flies through a certain waypoint at 4:00, the difference between that time and and the reference time is 16 hours. Thus, a forecast published on the first of January at 12:00 for 18 hours later will be selected for that waypoint.

Note that some flights may require forecasts of the day before the flight.

The files provided by ECMWF are processed in order to obtain the factor p_{risk}. The p_{risk} is computed applying the method described in [12], based on two parameters: the total totals index (TT) and the convective precipitation (CP). Both parameters TT and CP can be found in ECMWF high-resolution dataset, not available to the general public. Instead, the low-resolution dataset, which is public, is used.

From this dataset, TT can be computed as the sum of the vertical totals (VT) and the cross totals (CT) [12].

$$TT = VT + CT \tag{5}$$

$$VT = T(850\ hPa) - T(500\ hPa) \tag{6}$$

$$CT = T_{dew}(850\ hPa) - T(500\ hPa) \tag{7}$$

where $T(850\ hPa)$ denotes the temperature at a pressure altitude of 850 hPa.

According to table 1, with a value of TT of 44 $°C$ it is possible to form moderate thunderstorms. This is the threshold used for determining convective conditions. Since the data for CP is not available, this parameter is not taken into account in the analysis.

The probability is defined as the number of members of the ensemble which have a TT higher than the threshold value divided by the total number of members.

	Thunderstorm activity			
TT_i	Moderate	Heavy	Severe	Tornadoes
< 44	-	-	-	-
44-45	Isolated	-	-	-
46-47	Scattered	Few	-	-
48-49	Scattered	Few	Isolated	-
50-51	*	Scattered	Few	Isolated
52-55	*	Numerous	Few/scattered	Few
> 55	*	Numerous	Scattered	Scattered

Table 1: Operational taxonomy of risk of severe weather activity (Courtesy of [12])

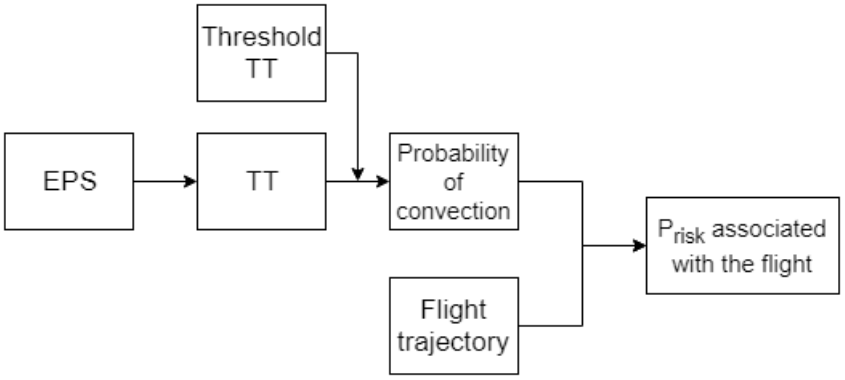


Figure 5: Scheme of p_{risk} computation.

In that way, each forecast consists on a 2D array in which for given equispaced values of longitude and latitude a probability of having convection is yielded. However, the value of the probability is required at any arbitrary point. Thus, a 2D interpolation is required. This is achieved using the interpolate function *RectBivariateSpline*, a 2D spline method from the library *SciPy*. A detailed description of this function is given in [18].

The p_{risk} of a trajectory is computed as the addition of the p_{risk} obtained in the segment between two consecutive waypoints. Each segment of length L is divided in N subsegments with a certain step Δ as follows:

$$N = int\left(\frac{L}{\Delta}\right) \tag{8}$$

Note that since N must be an integer number, the real step δx of the subsegments is smaller than or equal to Δ.

$$\delta x = \frac{L}{N} \tag{9}$$

For each division point of the segment, the probability of convection is computed by interpolation. The total probability I can be computed as a sum of rectangle quadratures:

$$I_{segment} = \int I\left(x(t)\right) dt \approx \sum_{i=0}^{N-1} I_{point,i} \cdot \delta x \tag{10}$$

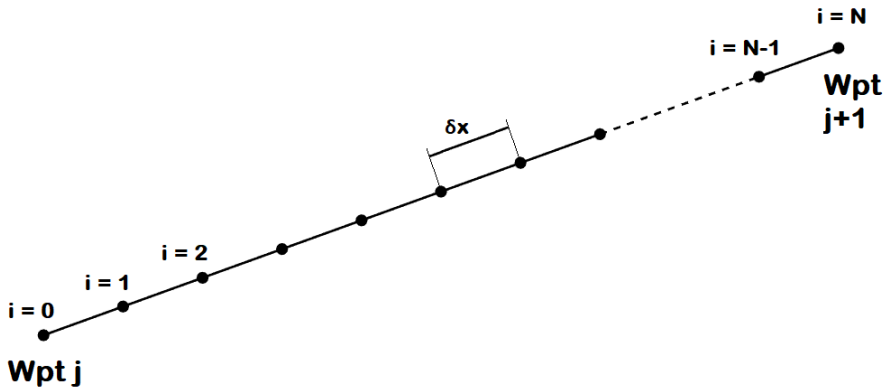


Figure 6: Sketch of divisions in a segment

Figure (6) shows schematically the divisions in a segment between two consecutive waypoints. Note that the last point $i = N$ of the segment is not considered in the estimation of the probability in the segment since it will be the first point of the next segment.

Since in equation 3 the distance is computed using the radius of the Earth in kilometers, the length of the segment is given in the same units. In equation 10 the probability is multiplied by the distance δx of the subsegment. Thus, the factor p_{risk} has units of kilometers.

Finally, the expression that gives the total p_{risk} accumulated in the intended trajectory divided in M segments is given by:

$$p_{risk} = \sum_{j=1}^{M} I_{segment,j} = \sum_{j=1}^{M} \left(\sum_{i=0}^{N-1} I_{point,i} \cdot \delta x \right)_{segment,j} \tag{11}$$

The factor p_{risk} is computed for the intended trajectory. The aim is to analyze the influence of this factor in the real trajectory. The next phase consists of generating a data frame which contains relevant information to be studied and performing the statistical analysis.

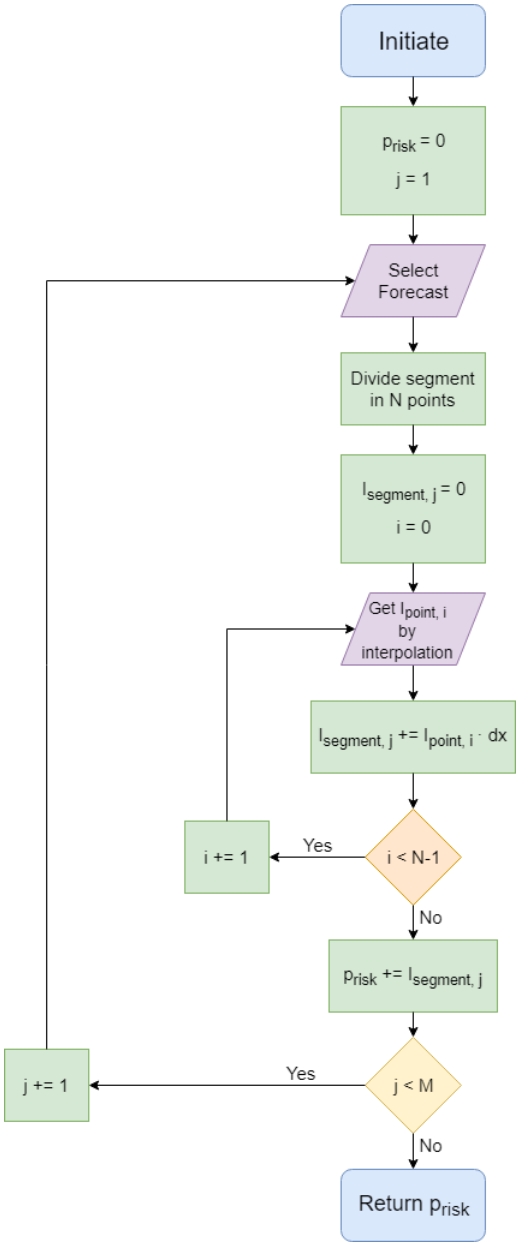


Figure 7: p_{risk} computation flowchart

3.3 Generation of data frame and statistical analysis

In this section it is explained how the data frame is generated and structured. There are two types of variables which are going to be correlated with the p_{risk}: delays and

variation of the intended trajectory (in terms of time and distance).

The route time (either estimated or actual) of the flight is computed as the difference between the landing and take-off times:

$$t_{route} = LDT - TOT \tag{12}$$

In order to be able to compare the effect of the weather conditions in the length of the trajectory of an aircraft between flights with different routes, the difference between the original route length filed in the flight plan and the real flown length will be normalized dividing by the original length. The same method will be applied to compare the difference in route time.

$$\Delta d(\%) = \frac{d_{real} - d_{estimated}}{d_{estimated}} \cdot 100 \tag{13}$$

$$\Delta t_{route}(\%) = \frac{t_{route,\ real} - t_{route,\ estimated}}{t_{route,\ estimated}} \cdot 100 \tag{14}$$

The delay is computed for three phases: of-block, take-off and landing. In the three cases, the delay is defined as the difference between the actual and the estimated times. Note that the delay might be negative in some cases.

These five variables, together with the p_{risk} and the aircraft WTC are stored in a *Python* dictionary, which is later transformed into a *Pandas* data frame. The structure of this data frame is shown in table 2.

Variable	**Type**	**Units**
p_{risk}	float	km
Variation of distance	float	%
Variation of time	float	%
Delay in Off-Block	float	min
Delay in Take-Off	float	min
Delay in Landing	float	min
Aircraft WTC	string	-

Table 2: Structure of data frame

The correlations used to connect the p_{risk} with the other variables are simple linear regressions. The results are displayed for each aircraft WTC category. In addition, results include histograms of all variables studied, in order to give an idea of the tendencies of the sample.

3.4 Code and limitations

In this section some aspects and considerations of the code are described. As stated before, the programming language used in the project is *Python*. The complete code can be found in the appendix, consisting on three scripts: the first script defines functions to extract the data of the ALL_FT+ files and store it in the *Flight* object list, as well as functions to compute the p_{risk}. The second script reads the ALL_FT+ files and using the functions previously defined, generates the data frame. Last but not least, the third script retrieves the data frame and produces the suitable correlations, displaying the results.

Previously, certain libraries have been mentioned. Table shows the complete list of libraries which have been used and their utility.

Library	**Utility**
Os	Obtaining the directory of the required files
Datetime	Managing dates and times of the flights and waypoints
SciPy	Used in the 2D interpolation of the prisk
NumPy	Defining N-dimensional arrays and working with them
Pandas	Generating and storing the data frame
Time	Obtaining the time of computation of the code
Sklearn	Obtaining the linear regressions to correlate prisk
Matplotlib	Displaying the results in plots

Table 3: *Python* libraries used in the code

The main limitation of the code is the time of computation of the p_{risk}. Thus, the optimization of the code will try to lessen the complexity and computations in the p_{risk} calculation.

Figure 7 showed the scheme to compute the risk of convection of a trajectory. Since for two different waypoints the forecast used may be different due to the difference in time, a forecast file needs to be opened for each segment. Constantly opening and closing files may slow down the computation.

In order to avoid opening each file multiple times, a class object *File_manager* is defined. It contains a dictionary where the forecast arrays are stored. A class method is defined to check if a forecast has been already stored. If so, it provides the data. If the forecast has not been loaded, it opens the file.

Other consideration in order to reduce the computation time is the length of the subsegments described in 3.2. The smaller the step, the more subsegments considered, and thus a higher accuracy is achieved. However, an excesive amount of subsegments would increase unnecessarily the computation time. In order to find an optimal value for the step, a sensitivity analysis is performed.

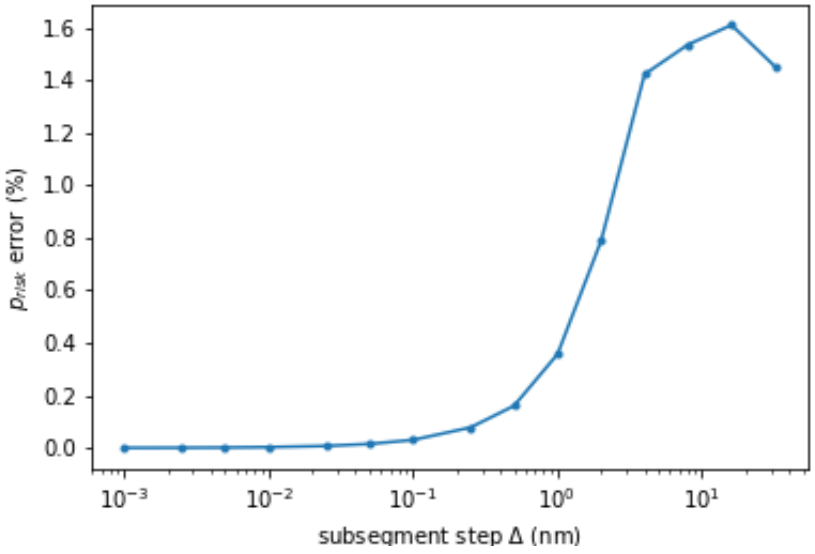


Figure 8: Error in p_{risk} for different steps

Figure 8 shows how the error in p_{risk} varies for a given trajectory when using different steps. In general the error is quite small (lower than 2%). The error using an arbitrary step Δ is calculated assuming that the p_{risk} obtained for a step of 0.001 nm is the exact value, according to equation 15.

$$\epsilon(\%) = \frac{p_{risk}(\Delta) - p_{risk}}{p_{risk}} \cdot 100 \tag{15}$$

In order to account for the computation time, table 4 shows the time required to analyze one flight. This time was obtained by running the code with a small sample (750 flights) and different steps. ALL_FT+ files have between 20000 and 35000 flights. The time of computation required for 30000 flights is also reflected in the table in order to give an idea of the total time required to generate the data frame of one file (one day of flight data).

Step (nm)	**Time per flight (s)**	**Time per 30000 flights (h)**	**Error (%)**
5	0.93643	7.8036	1.4263
2	0.94256	7.8547	0.7901
1	0.95463	7.9553	0.3641
0.5	0.95715	7.9762	0.1661
0.25	0.96041	8.0034	0.0751
0.1	0.96327	8.0272	0.0377

Table 4: Computation time for different steps

It is seen that the difference in the computation time is not so substantial. The reason might be that task which requires more computation time is the reading of the files and not the computation itself. Therefore, a high resolution step can be selected. A step of 0.25 nm is chosen, yielding a solution with an error lower than 0.1%. Nevertheless, the time is still quite high.

The fact that the computation time is too high for one day, together with the DDR2 download policy led to the decision of reducing the sample. Only four months will be considered: January, February, May and June. Still, considering that one day requires 7-8 hours of computation time, the sample needs a higher reduction.

It was decided to consider only flights from the airline *British Airways*, which operates about 700-800 flights per day with destinations all around the globe. However, this airline does not have 'light' aircraft in its fleet. In order to take account of this type of aircraft 25 % random flights of this category (around 100 flights) were selected for each day analyzed.

In this way, about 900 flights were analyzed per day. Thus, the calculation considering a month of the sample would last, according to table 4, slightly less than 8 hours.

It was decided to study the data of each month separately, to avoid a long time of continuous computation. Hence, the code should be adapted to be able to generate the data frame in more than one run. To that aim, after generating the data frame the code checks if there is an existing file. If this is the case, the previous data frame is opened and combined with the new one.

4 Results and discussions

Figures 9 to 11 show the effect of the parameter p_{risk} in the delay in off-block for the three aircraft categories. Since the linear regressions have positive slope, a certain relationship could be established between this delay and the factor of risk.

However, this relation is not as pronounced as it was expected. Flights with substantial delays are found in regions of low factor of risk. The reason might be that the majority of these delays are produced due to different causes other than exposure to convection.

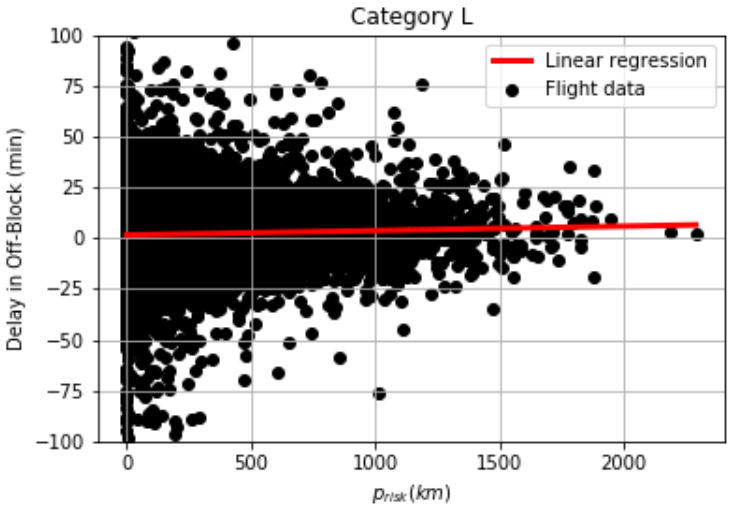


Figure 9: Effect of p_{risk} in delay in Off-Block (Light aircraft)

It is significant that in the case of light aircraft (figure 9) the shape is nearly symmetric. Light aircraft are mostly used as private jets and in general aviation. Therefore they are not subjected to commercial schedules, and may depart earlier if authorized.

Most of these early departures are found in a low risk region, which could be an indicator of the effect of p_{risk}: at low convective risk, early departures are authorized. Nevertheless, the majority of delays are also found in the low risk region, making this dependency uncertain.

In the other two categories, the flights are subjected to commercial schedules and departing earlier is not so frequent. Despite this, if the boarding process is completed before schedule, a flight may request an earlier slot, and a negative value of the delay is obtained. In addition, particular cases might reschedule the departure to an earlier time.

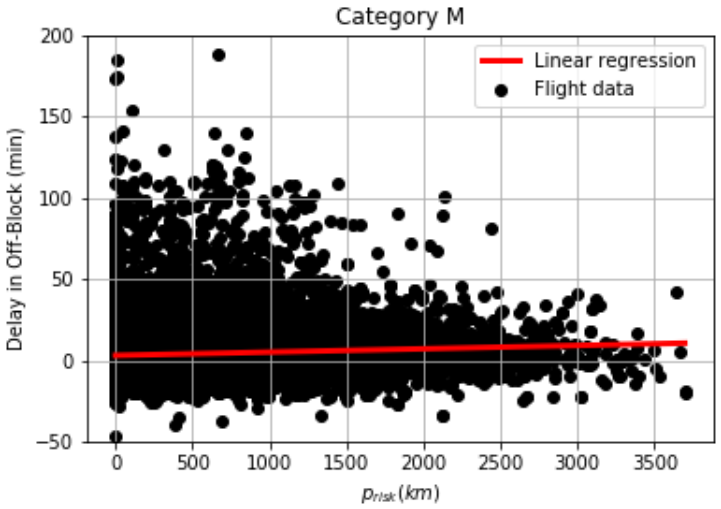


Figure 10: Effect of p_{risk} in delay in Off-Block (Medium aircraft)

It appears that the largest dependency in the delay with the factor of risk is given in the medium aircraft, being the slope more pronounced.

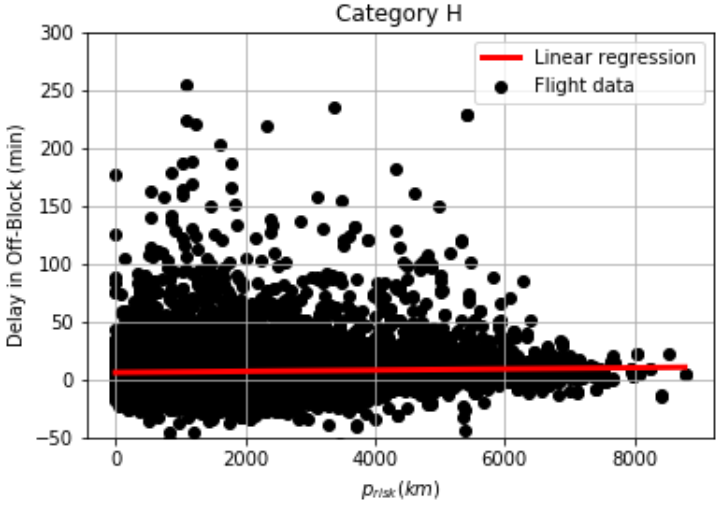


Figure 11: Effect of p_{risk} in delay in Off-Block (Heavy aircraft)

It can be observed that the larger the aircraft, the higher the maximum value of p_{risk}. This makes sense since a longer route may be more exposed to convective regions. The most significant delays are found in heavy aircraft, reaching a peak over 250 minutes, since a longer route is more subjected to this type of uncertainty.

The effect in the take-off delay is very similar to the previously studied. The difference between the take-off and the off-block time is the taxi time. If the delay in

take-off is higher than the delay in off-block, the aircraft might have been retained in the ramp waiting for an authorization to take-off.

The dependencies for the three categories appear to be the same as for the off-block delay. This proves that the value of the p_{risk} does not affect the taxi time.

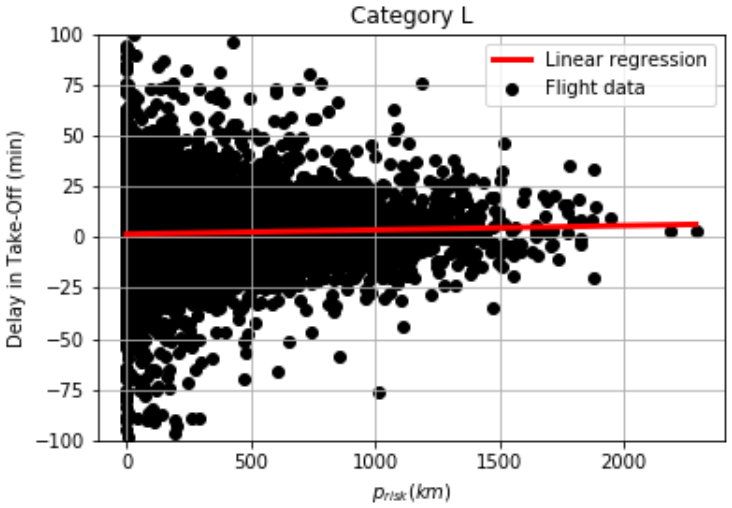


Figure 12: Effect of p_{risk} in delay in Take-Off (Light aircraft)

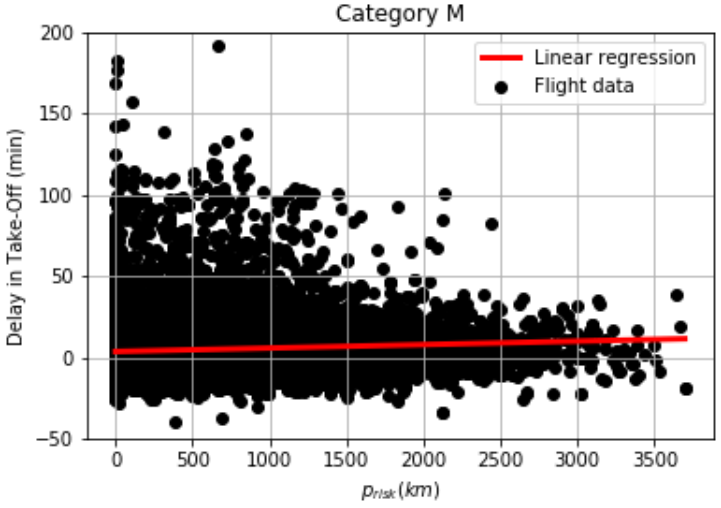


Figure 13: Effect of p_{risk} in delay in Take-Off (Medium aircraft)

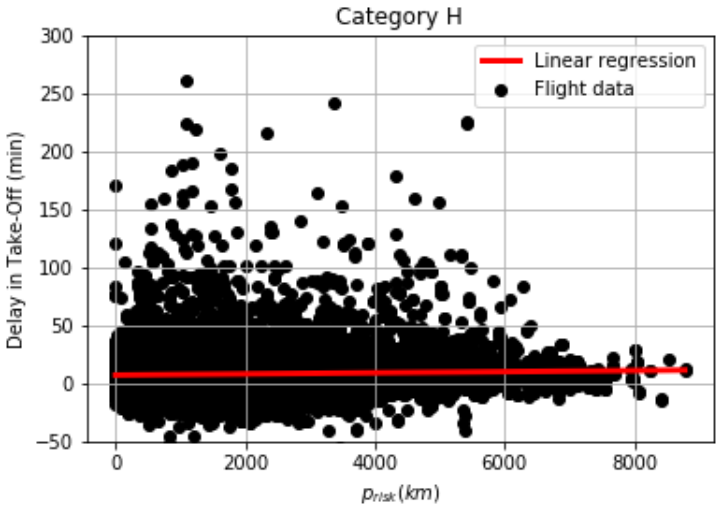


Figure 14: Effect of p_{risk} in delay in Take-Off (Heavy aircraft)

Looking at the delay in landing, it is appreciated that in general the behaviour remains similar. The main difference is that the peaks in delays are higher in landing than in the departure delays. Flights with a significant increase in this delay may have suffered several changes in the route in order to avoid convective regions, or in particular cases a change of destination.

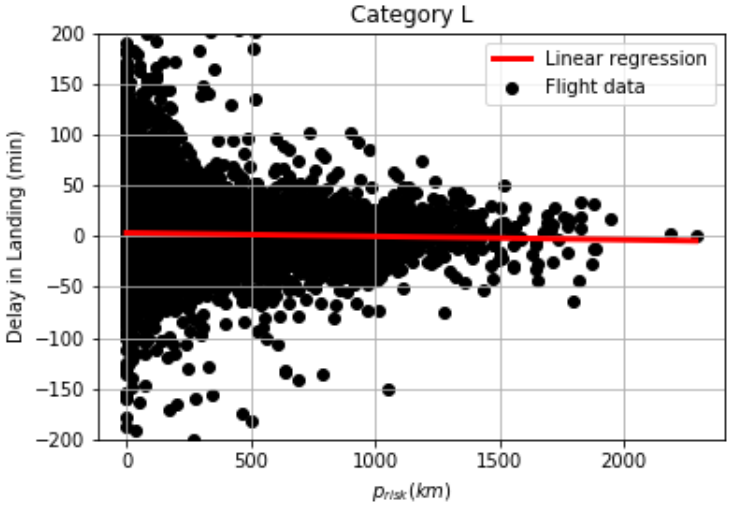


Figure 15: Effect of p_{risk} in delay in Landing (Light aircraft)

In light aircraft, the delay in landing slightly decreases with p_{risk}. In the same way as this type of aircraft had more flexibility in the departure, they are also flexible in the arrival. In the other two categories the delay increases with the factor of risk.

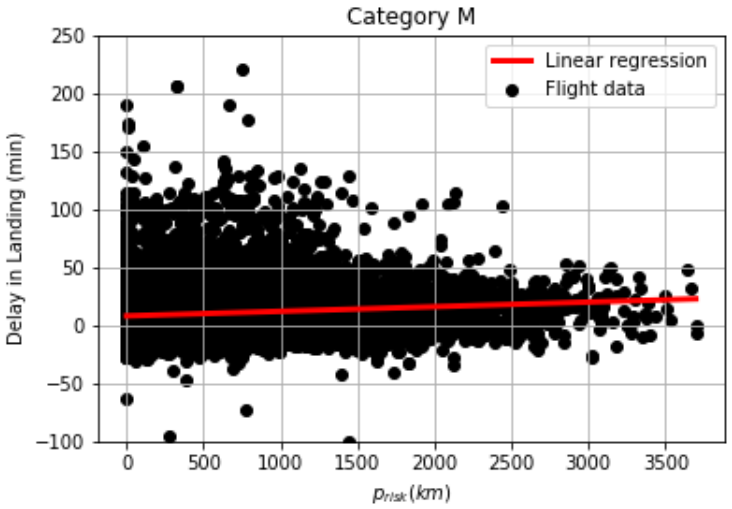


Figure 16: Effect of p_{risk} in delay in Landing (Medium aircraft)

In commercial aviation, aircraft carriers want to decrease the delay at arrival, by terms of flying faster (whenever is possible) when the flight has departed late. Thus, the values of the delay at landing tend to be smaller than in the departure, although there are special cases in which the total delay has been increased (peak values).

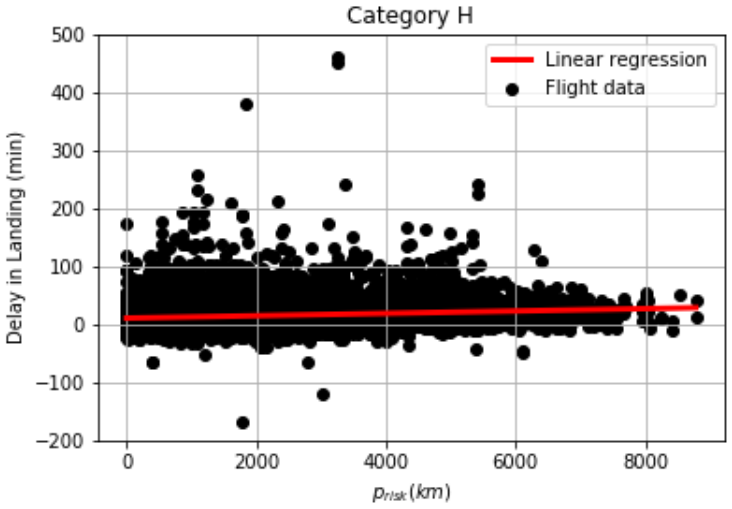


Figure 17: Effect of p_{risk} in delay in Landing (Heavy aircraft)

The following figures (18 - 20) represent the effect of the p_{risk} in the variation in total length of the route. The minimum value for this parameter is a variation of -100%, meaning that the real length of the route was 0 km. This value may be associated to cancelled flights. It is expected that a higher exposure to convection causes more changes in the route, generally increasing the travelled distance.

However, the most significant variations in length are given at low p_{risk}. This means that the exposure to convective regions is not the main cause of variation in distance. For instance, a change of destination (a non frequent event) changes notably the distance travelled.

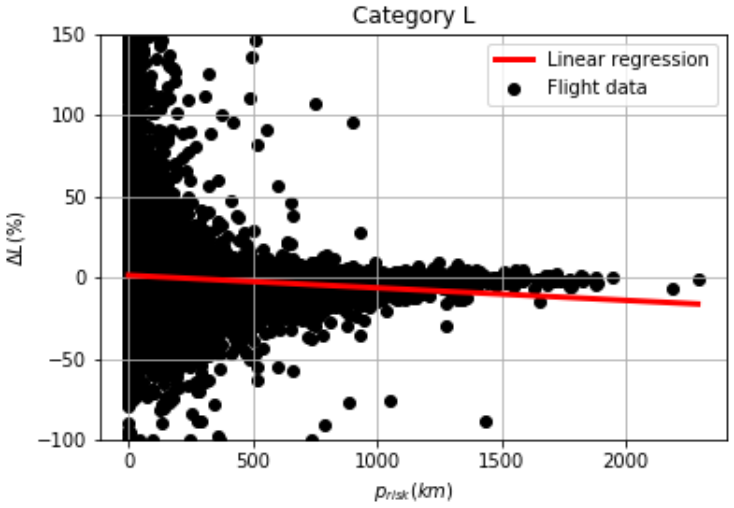


Figure 18: Effect of p_{risk} in ΔL (Light aircraft)

It is seen that in the case of light aircraft, for low exposure to convection, there is a significant number of flights having a considerable decrease in the route length, as well as several flights increasing the travelled distance. Once again, the reason is the flexibility associated to this type of flights, allowing changes in the destination, which result in an important variation in the length. There are also several flights near the minimum value of -100%. Also, this type of aircraft have relatively short routes, and any change in the trajectory would affect significantly this parameter.

The relation of this parameter with the factor of risk is negative in this category, since it decreases. The high amount of flights increasing the route length when the p_{risk} is low might alter the correlations. This category of aircraft comprises leisure flights where the users may spend more time in the air for their enjoyment. It is significant that this increase in travelled distance is given at low exposure to convection.

In medium and heavy aircraft, changes in routes in normal conditions (no change of destination) consist only on small corrections to avoid convective regions as well as ATM clearances to avoid certain conflicts. Thus, the change in travelled distance is smaller.

In the case of medium aircraft the correlation line is an horizontal line, not showing a dependency of these two factors. For heavy aircraft the line is slightly positive sloped, but the relation is not clear.

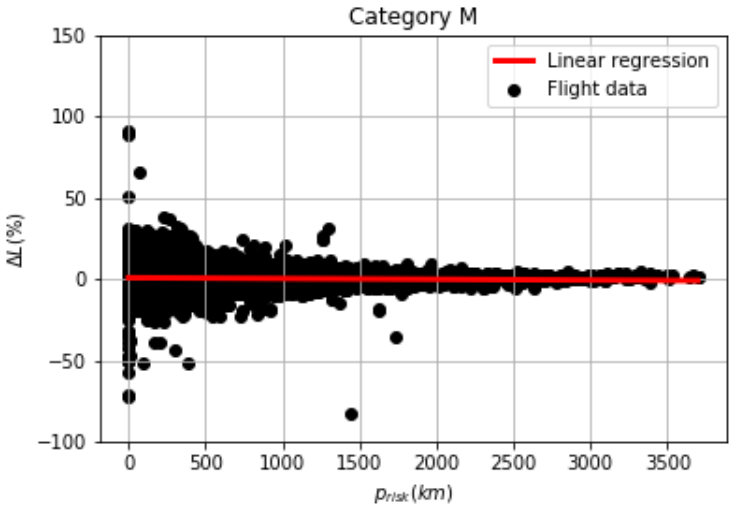


Figure 19: Effect of p_{risk} in ΔL (Medium aircraft)

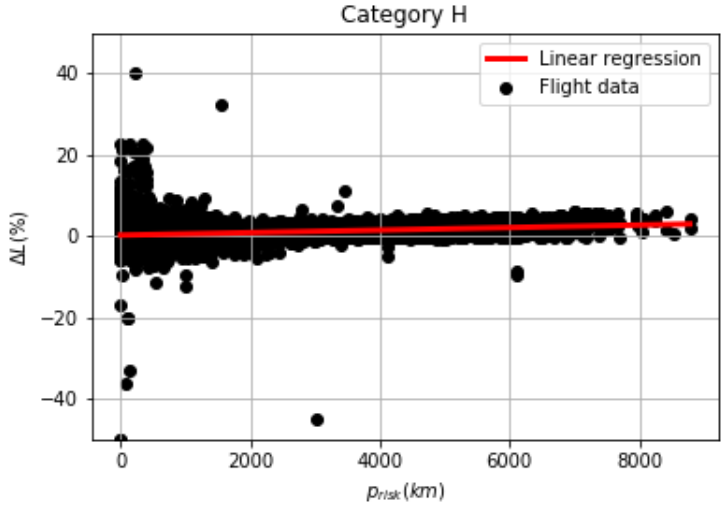


Figure 20: Effect of p_{risk} in ΔL (Heavy aircraft)

The expectations in the variation in time are similar to the ones in variation in distance. A higher exposure to convection would provoke more deviations, which should increase the time of the flight. However, these deviations can be compensated with an increase in velocity.

In figure 21 it is seen that the flexibility of private aircraft is even higher in terms of time than distance. As it happened in the case of the variation in distance, the time flown in this type of aircraft tends to increase when there is no risk of convection. Still, the large amount of flights having such a significant variation in distance and time is rare.

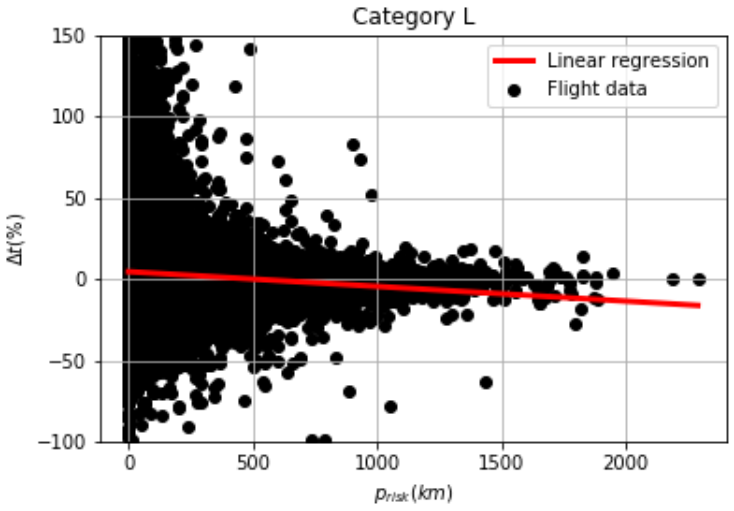


Figure 21: Effect of p_{risk} in Δt (Light aircraft)

In the case of light aircraft the variation in time seems to decrease with the factor of risk for the same reasons as the variation of distance did.

For medium aircraft, this parameter also seems to decrease, whereas for heavy aircraft the linear regression line is an horizontal line. Therefore a dependency of the factor of risk and the variation in time cannot be established. As stated before, the flight time depends also on the velocity, which is a parameter not studied in this project. Therefore, it is difficult to find a relation for the time.

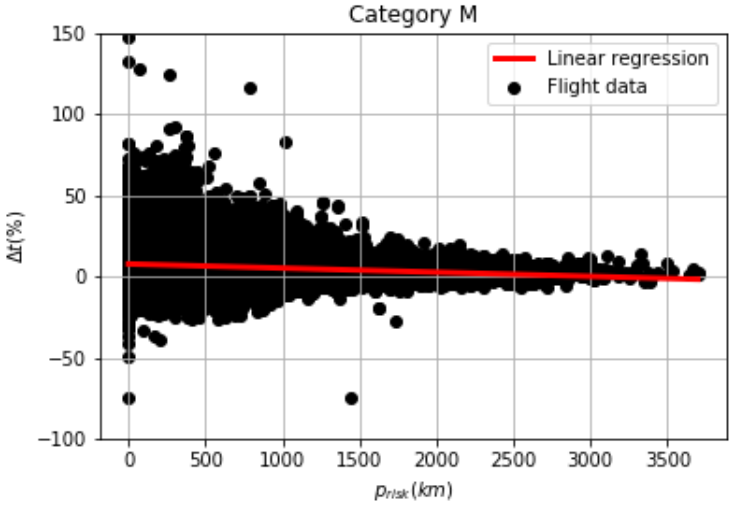


Figure 22: Effect of p_{risk} in Δt (Medium aircraft)

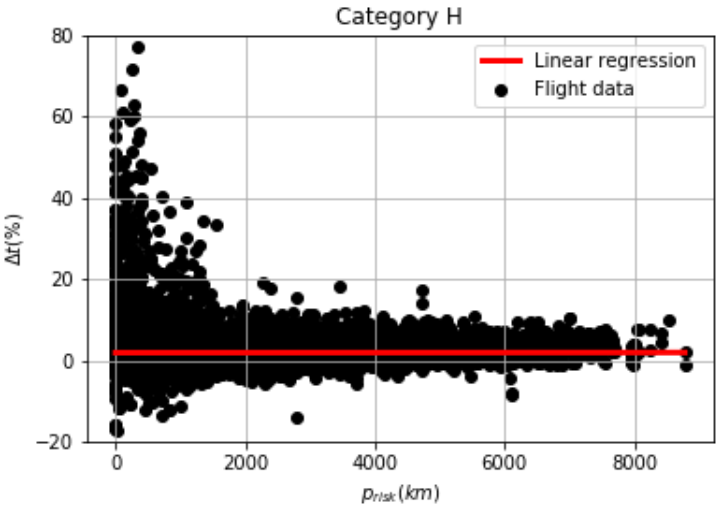


Figure 23: Effect of p_{risk} in Δt (Heavy aircraft)

Tables 5 and 6 show the variance and the standard deviation of the parameters studied, as an indicator of spread of the sample. It is seen that the spread is large in general and particularly in the p_{risk}. Other important remark is that the deviation is higher in medium aircraft for delays in off-block and take-off, whereas for delays in landing and variations of distance and time, the deviation is higher in light aircraft. The deviation is maximum in the factor of risk for heavy aircraft, since this category has a wider range of values.

Variable	**Category L**	**Category M**	**Category H**
Delay in OB	223.9	418.4	271.9
Delay in TO	224.3	424.0	280.2
Delay in LD	794.0	549.3	449.2
ΔL	398.8	24.1	4.9
Δt	579.8	130.2	14.8
p_{risk}	71874.1	275646.8	1903715

Table 5: Variance of the variables studied

Variable	**Category L**	**Category M**	**Category H**
Delay in OB	14.96	20.46	16.49
Delay in TO	14.98	20.59	16.74
Delay in LD	28.18	23.44	21.19
ΔL	19.97	4.91	2.20
Δt	24.08	11.41	3.85
p_{risk}	268.09	525.02	1379.75

Table 6: Standard deviation of the variables studied

Finally, tables 7 to 9 indicate the linear regression parameters ($y = b_0 + b_1 \cdot x$) of the correlations showed in the figures for the three categories. These parameters are calculated using a 95% confidence interval.

Variable	b_0	$b_1 \cdot 10^3$
Delay in OB	1.553 ± 0.258	1.265 ± 0.783
Delay in TO	1.449 ± 0.258	2.112 ± 0.783
Delay in LD	3.282 ± 0.486	-3.441 ± 1.474
ΔL	1.211 ± 0.342	-7.712 ± 1.040
Δt	4.420 ± 0.413	-9.059 ± 1.254

Table 7: Linear regression parameters for light aircraft

Variable	b_0	$b_1 \cdot 10^3$
Delay in OB	2.953 ± 0.329	2.040 ± 0.432
Delay in TO	3.524 ± 0.331	2.116 ± 0.434
Delay in LD	8.116 ± 0.375	3.966 ± 0.493
ΔL	0.688 ± 0.079	-0.479 ± 0.104
Δt	7.588 ± 0.182	-2.548 ± 0.239

Table 8: Linear regression parameters for medium aircraft

Variable	b_0	$b_1 \cdot 10^3$
Delay in OB	6.234 ± 0.432	0.513 ± 0.165
Delay in TO	7.263 ± 0.439	0.482 ± 0.168
Delay in LD	10.740 ± 0.551	2.037 ± 0.211
ΔL	0.238 ± 0.057	0.312 ± 0.022
Δt	1.858 ± 0.101	0.005 ± 0.039

Table 9: Linear regression parameters for heavy aircraft

It is statistically significant that for the delays - except for the delay in landing in light aircraft - the the slope is positive in the confidence interval (0 is not in the interval). Moreover, it is seen that the slope for the variation in time is smaller than the slope for the variation in distance. This could represent an average acceleration of the aircraft when they vary the trajectory, which would result in an increase of the fuel consumed.

There are many factors which can cause a delay, making difficult to find a notable dependency of the parameters studied. In addition, the results might differ if CP had been taken into account since there can be flights having a risk of exposure to convective precipitation which has not been considered.

5 Regulatory framework

Given the fact that this is a research project there is no legislation applicable to the development of the methods described.

Files from DDR2 and ECMWF (TIGGE) are available for academic and research purposes. This project complies with their corresponding license agreements. The author and the tutors confirm they have obtained permission from the copyright holder of any third party material included in their project.

All the computations and simulations have been carried out following Python programming standards.

6 Socio-economic environment

In this section, the aspects related with the economic and social environment of the project are discussed. The budget of the project is estimated considering the different resources used. Also, the impact of the project is analyzed.

6.1 Project budget

The total budget corresponding to this project can be roughly estimamted as follows:

- **Worker salary:** According to [19], the annual salary of a research engineer with less than one year of experience in Spain is 16933 €. This quantity corresponds to an income per hour of 8.14 €, assuming a standard working day of eight hours. Considering a total time dedicated to the project of 400 hours, the total salary is 3256 €.
- **Equipment:** The computations were implemented on a personal computer *HP* ® *Pavilion* with an intel ® CORE^{TM} i7-4500U processor, and 8 *GB* of RAM. This computer is currently valued at about 600 €. Also, a 2 *TB* external drive was used to store all the files used, valued in 70 €.

Since *Python* is an open source language there is no budget designated to programming licenses.

Additionally, the project could be improved by subscribing to MARS, which allows access to ECMWF ENS dataset. In this way the project could account for CP analysis, and there would not be necessity to compute the parameter TT, since it is already given in ENS. This would mean an increase of the accuracy of the probability of convection. An annual subscription to MARS costs 5000 GBP.

Without considering this additional expense, the total budget of the project is then 3926 €.

6.2 Socio-economic impact

The results obtained in this project could contribute to a better understanding of the effects of the convective regions uncertainties in the aircraft trajectories and delays. The correlations obtained could be used in software dedicated to ATM performance to optimize the decision-making at pre-tactical level. This could increase the predictability of the ATM system, leading to an improvement in the high-level goals: efficiency, capacity, safety and environmental impact.

The uncertainty of capacity would decrease since the effect of the weather uncertainties are better understood. Therefore the capacity increases. In addition, the safety would increase since the adequate separation could be provided between aircraft. The decrease in uncertainty would also reduce the environmental impact. The trajectories would have less variations and contingency fuel would be saved, reducing the pollution to atmosphere.

Finally, an optimization in decision-making at pre-tactical level could decrease the delays of the flights. Nowadays the society associates airports with long queues and delays, which lowers the user perception of the air transport. If this project helps decreasing the delays of the flights, it will contribute to an amelioration of the user perception.

7 Conclusion

To sum up, this project tried to identify the cause-effect relationship between the exposure to convection and the changes in the planned trajectory. To that aim, the required flight data (departure times, point profile trajectories, aircraft operator, etc.) was gathered from Eurocontrol DDR2 database. On the other hand, flight forecasts were requested to ECMWF TIGGE, to obtain the regions with risk of convection.

With these two sets of data, a factor of risk was associated to each flight, in order to try to establish a relation between this factor and the changes in the trajectory. The flight variables studied were the delays in off-block, take-off and landing, the variation in the route length and the variation in flown time. The results were separated according to the aircraft Wake Turbulence Category.

Some tendencies could be identified, such as the flexibility associated to private flights, which are subjected to commercial schedules and may depart at a different time if allowed, and spend more time in the air. However, since light aircraft are mainly used in general aviation, they do not represent commercial aviation. More significant for the scope of the project was identifying a slight increase of the delays with the factor of risk in medium and heavy aircraft could be appreciated.

In general, the dependencies found are not notably strong, but considering that the factors influencing the trajectories and delays of a flight are many, the tendencies found are statistically significant. The results of this project explain a small part of the generation of delays and changes in trajectory, and the factor of risk can be remarked as a significant predictor.

The fact that the convective precipitation was not considered in the analysis could have been relevant in the results. Furthermore, flights delayed for reasons different from weather conditions may affect the results.

The results of this project are a first step to quantify the cost of convective conditions, and they can be employed in decision-making at a pre-tactical level.

In future projects, this procedure could be used to obtain a better understanding of the effect of this factor of risk. Further considerations for the future should account for the convective precipitation, as an additional parameter for the risk of convection, as well as a filter to only analyze flights delayed by weather conditions. A supplemental study would be the analysis of the wind uncertainty.

References

[1] Charles A. Doswell III. *Severe Convective Storms*. National Severe Storms Laboratory. Norman, Oklahoma 73069, May 2000.

[2] Beth Krajewski. *Flying in Convective Weather... And Why You Shouldn't*. [Online]. Available: `https://business.weather.com/blog/flying-in-convective-weather-and-why-you-shouldnt`, accessed April 10, 2018. Sept. 2015.

[3] *Eurocontrol website*. [Online]. Available: `http://www.eurocontrol.int/`, accessed February 24, 2018.

[4] Enrique Casado, Miguel Vilaplana, and Colin Goodchild. "Sensitivity of trajectory prediction accuracy to aircraft performance uncertainty". In: *AIAA Infotech@ Aerospace (I@ A) Conference*. 2013, p. 5045.

[5] Rafael Vazquez and Damián Rivas. "Propagation of initial mass uncertainty in aircraft cruise flight". In: *Journal of Guidance, Control, and Dynamics* 36.2 (2013), pp. 415–429.

[6] Enrique Casado, Miguel Vilaplana, and Colin Goodchild. "Sensitivity of continuous climb departure predictions to aircraft intent uncertainties". In: *ATACCS'2013* (2013), p. 202.

[7] *SESAR Joint Undertaking website*. [Online]. Available: `https://www.sesarju.eu/`, accessed May 30, 2018.

[8] Peter Bauer, Alan Thorpe, and Gilbert Brunet. "The quiet revolution of numerical weather prediction". In: *Nature* 525.7567 (2015), p. 47.

[9] Jacob Cheung et al. "Recommendations on trajectory selection in flight planning based on weather uncertainty". In: *Proc. 5th SESAR Innovation Days (SID2015), Bologna, Italy* (2015), pp. 1–8.

[10] M. Sanjurjo D. González-Arribas M. Soler. *Wind-Based Robust Trajectory Optimization using Meteorological Ensemble Probabilistic Forecasts*. 6th SESAR Innovation Days. Nov. 2016.

[11] M. Sanjurjo D. González-Arribas M. Soler. "Robust Aircraft Trajectory Planning Under Wind Uncertainty Using Optimal Control". In: *Journal of Guidance, Control, and Dynamics* (Oct. 2017).

[12] J. García-Heras D. González-Arribas M. Soler. *Robust Optimal Trajectory Planning under Uncertain Winds and Convective Risk*. 5th ENRI International Workshop on ATM/CNS. Nov. 2017.

[13] *DDR2 Reference Manual For General Users*. Eurocontrol.

[14] Roger W Sinnott. "Virtues of the Haversine". In: *Sky Telesc.* 68 (1984), p. 159.

[15] *ICAO. DOC 8643 Aircraft Type Designators*. [Online]. Available: `https://www.icao.int/publications/DOC8643/Pages/Search.aspx`, accessed May 17, 2018.

[16] *ICAO Flightplan Form Basics*. EuroFPL. 2012.

[17] *ECMWF website*. [Online]. Available: `https://www.ecmwf.int/`, accessed May 23, 2018.

[18] *SciPy website*. [Online]. Available: `https://docs.scipy.org/doc/scipy/reference/index.html`, accessed March 22, 2018.

[19] Aitor Díaz Lucas. *Encuesta de Salarios y Actividad profesional*. [Online]. Available: `http://www.ingeniariak.eus/wp-content/uploads/2017/04/Encuesta-salarios-Ingenieros-Industriales-2016-2017.pdf`, accessed June 10, 2018. 2017.

Appendix A - List of acronyms

Term	Definition
ALDT	Actual Landing Time
AOBT	Actual Off-Block Time
ATM	Air Traffic Management
ATOT	Actual Take-Off Time
CP	Convective Precipitation
CT	Cross Totals
CTFM	Current Tactical Flight Model
DDR2	Demand Data Repository
ECMWF	European Centre for Medium-Range Weather Forecasts
ELDT	Estimated Landing Time
EOBT	Estimated Off-Block Time
EPS	Ensemble Prediction Systems
ETOT	Estimated Take-Off Time
FTFM	Filed Tactical Flight Model
ICAO	International Civil Aviation Organization
MARS	Meteorological Archival and Retrieval System
NWP	Numerical Weather Prediction
SESAR	Single European Sky ATM Research
TBO	Trajectory-Based Operations
TT	Total Totals Index
UC3M	Universidad Carlos III de Madrid
VT	Vertical Totals
WTC	Wake Turbulence Category

Appendix B - Code

This appendix shows the code used to obtain the results presented within this project. The whole code was implemented using *Python* through the computing environment *Spyder*. It is composed of three scripts.

ALLFT2FLIGHT.py

This script defines Aircraft and Wpt class objects, as well as some functions used to compute the different variables. Class Wpt is defined using a class method, from the data contained in a string. The class Flight contains a class method which computes the p_{risk} of the flight. This class method used a defined function to compute the p_{risk} of each segment.

Another class object is defined, which is the File_manager object. It is used to check if a file has been already opened and stores its data.

The function string2datetime transforms the dates in format YYYYMMDDhhmmss into an object of the class datetime. The function distance computes the length joining two point in the surface of the Earth given their coordinates. Finally, the function select_forecast determines the suitable forecast file for each segment.

```
# -*- coding: utf-8 -*-
"""
Created on Mon Mar  5 11:17:51 2018

@author: Juan Rodriguez Carrillo

TFG: Trajectory predictability analysis under convective weather conditions

ALLFT2FLIGHT.py defines class objects to store data to be read from ALL_FT+
files
"""

import os
import datetime
from math import sin,cos,sqrt,radians,atan2
from scipy import interpolate
import numpy as np


"""
Define Waypoint class
"""

class Wpt(object):
    def __init__(self,time,lat,lon,name,lvl):
        self.time = time
        self.lat = lat
        self.lon = lon
        self.name = name
        self.lvl = lvl


    @classmethod
    def create_from_string(cls,s):
        divisions = s.split(":")
```

```
        if divisions[6]:
            t = divisions[0] #Time (string)
            time = string2datetime(t[0:14]) #Time (datetime)
            name = divisions[1] #Name of the waypoint
            lvl = divisions[3] #FLight Level
            coord = divisions[6] #Coordinates
            lat1 = coord[0:6] #Latitude (string)
            lon1 = coord[7:14] #Longitude (string)
            lat = float(lat1[0:2]) + float(lat1[2:4])/60 +\
            float(lat1[4:6])/3600 #Latitude in degrees (float)
            lon = float(lon1[0:3]) + float(lon1[3:5])/60 +\
            float(lon1[5:7])/3600 #Longitude in degrees (float)
            if coord[6] == 'S':
                lat = -lat #Negative for South
            if coord[14] == 'W':
                lon = -lon #Negative for West
        else: #In case empty waypoint
            time = ''
            lat = ''
            lon = ''
            name = divisions[1]
            lvl = ''

        return cls(time,lat,lon,name,lvl)


"""
Define Flight class
"""

class Flight(object):
    def __init__(self,dep_aero,arr_aero,AC_type,AC_cat,eobt,\
                 aobt,filed_wpts,current_wpts):
        self.dep_aero = dep_aero #Departure aerodrome
        self.arr_aero = arr_aero #Arrival aerodrome
        self.AC_type = AC_type #Aircraft type
        self.AC_cat = AC_cat #Aircraft category
        self.eobt = string2datetime(eobt) #EOBT
        self.aobt = string2datetime(aobt) #AOBT
        self.filed_wpts = filed_wpts #Estimated waypoints
        self.current_wpts = current_wpts #Actual waypoints
        self.prisk = 0 #Factor of risk
        filed_L = 0 #Initialize planned route length
        current_L = 0 #Initialize actual route length
        for i in range(len(self.filed_wpts)-1):
            if (self.filed_wpts[i].name != 'ZZZZ') and\
            (self.filed_wpts[i+1].name != 'ZZZZ'):
                dL = distance(self.filed_wpts[i].lat,self.filed_wpts[i+1].lat,\
                              self.filed_wpts[i].lon,self.filed_wpts[i+1].lon)
                filed_L += dL
        for j in range(len(self.current_wpts)-1):
            if (self.current_wpts[j].name != 'ZZZZ') and\
            (self.current_wpts[j+1].name != 'ZZZZ'):
                dL2 = distance(self.current_wpts[j].lat,\
                               self.current_wpts[j+1].lat,\
                               self.current_wpts[j].lon,\
                               self.current_wpts[j+1].lon)
                current_L += dL2
        self.filed_distance = filed_L #Estimated Route Length (km)
        self.current_distance = current_L #Actual Route Length (km)
        if self.filed_wpts[0].name != 'ZZZZ':
            etot = self.filed_wpts[0].time #Estimated Take-Off Time
        else:
            etot = self.filed_wpts[1].time #Estimated Take-Off Time
        self.etot = etot
        if self.current_wpts[0].name != 'ZZZZ':
            atot = self.current_wpts[0].time #Actual Take-Off Time
```

```
        else:
            atot = self.current_wpts[1].time #Actual Take-Off Time
        self.atot = atot
        if self.filed_wpts[len(self.filed_wpts)-1].name != 'ZZZZ':
            elt = \
            self.filed_wpts[len(self.filed_wpts)-1].time #Estimated Landing Time
        else:
            elt = \
            self.filed_wpts[len(self.filed_wpts)-2].time #Estimated Landing Time
        self.elt = elt
        if self.current_wpts[len(self.current_wpts)-1].name != 'ZZZZ':
            alt = \
            self.current_wpts[len(self.current_wpts)-1].time #Actual Landing Time
        else:
            alt = \
            self.current_wpts[len(self.current_wpts)-2].time #Actual Landing Time
        self.alt = alt

    def compute_convection(self,file_manager,step_nm=5):
        #Prisk with estimated trajectory
        I = 0 #Initialize probability of having convection on the flight
        for i in range(len(self.filed_wpts)-1):
            if (self.filed_wpts[i].name != 'ZZZZ') and \
            (self.filed_wpts[i+1].name != 'ZZZZ'):
                I_segment = \
                compute_convection_segment(self.filed_wpts[i],\
                                           self.filed_wpts[i+1],\
                                           self.eobt,file_manager,step_nm)
                I += I_segment
        self.prisk = I


class File_manager(object):
    """
    This class manages prisk files, opening them and saving in a dictionary
    """
    def __init__(self):
        self.files_dict = {}

    def select_file(self,name_file):
        #name_file must be a string
        if name_file in self.files_dict:
            return self.files_dict[name_file]
        else:
            script_dir = os.getcwd()
            pathPriskFile = os.path.join(script_dir,'prisk',name_file)
            data = np.load(pathPriskFile)
            self.files_dict[name_file] = data
            return data



def compute_convection_segment(wp1,wp2,eobt,file_manager,step_nm=2):
    I_segment = 0 #Initialize probability of having convection on the segment
    file_prisk = select_forecast(wp1,eobt)
    data = file_manager.select_file(file_prisk)
    lats_conv = data['latitude']
    lons_conv = data['longitude']
    nlons_conv = np.unique(lons_conv).shape[0]
    nlats_conv = np.unique(lats_conv).shape[0]
    #npoints_conv = nlons_conv*nlats_conv
    new_shape = (nlats_conv, nlons_conv)
    lats_conv_plot = lats_conv.reshape(new_shape)
    lons_conv_plot = lons_conv.reshape(new_shape)
    prisk = data['variable'].reshape(new_shape)
    interpolator = interpolate.RectBivariateSpline(lats_conv_plot[::-1,0],\
                                                   lons_conv_plot[0,:],\
```

```
                                                    prisk[::-1,:], kx=1, ky=1)
    delta_lat = wp2.lat - wp1.lat
    delta_lon = wp2.lon - wp1.lon
    L = distance(wp1.lat,wp2.lat,\
                 wp1.lon,wp2.lon) #Distance between the two waypoints (km)
    N = int(L/(step_nm*1.852)) #Number of subsegments in the segment
    if N == 0:
        N = 1
    dlat = delta_lat/N
    dlon = delta_lon/N
    dx = L/float(N)
    for i in range(N):
        point_lat = wp1.lat + dlat*i
        point_lon = wp1.lon + dlon*i
        I_point = interpolator(point_lat,point_lon)
        I_segment += I_point[0,0]*dx
    return I_segment

def distance(lat1,lat2,lon1,lon2):
    delta_lat = lat2 - lat1
    delta_lon = lon2 - lon1
    r = 6371 #Radius of Earth (km)
    #Haversine formula
    a = (sin(radians(delta_lat)/2))**2 +\
    cos(radians(lat1))*cos(radians(lat2))*(sin(radians(delta_lon)/2))**2
    c = 2*atan2(sqrt(a),sqrt(1-a))
    L = r*c
    return L

def string2datetime(str_datetime):
    YYYY = int(str_datetime[0:4])
    MM = int(str_datetime[4:6])
    DD = int(str_datetime[6:8])
    hh = int(str_datetime[8:10])
    mm = int(str_datetime[10:12])
    ss = int(str_datetime[12:14])
    return datetime.datetime(YYYY,MM,DD,hh,mm,ss)

def select_forecast(wp,eobt):
    t_wp = wp.time
    t_ob = eobt
    t0 = t_ob - datetime.timedelta(hours = 3)
    #t0 is the time at which the flight plan is filed
    if t0.hour < 12:
        t_ref = datetime.datetime(t0.year,t0.month,t0.day,0,\
                                  t0.minute,t0.second)
        hour = '00'
    else:
        t_ref = datetime.datetime(t0.year,t0.month,t0.day,12,\
                                  t0.minute,t0.second)
        hour = '12'
    #tref is the time of the most recent available forecast

    delta_t = t_wp - t_ref
    delta_t = delta_t.days*24 + float(delta_t.seconds)/3600 #(hours)
    dt = 6*int((delta_t+3)/6) #Time step of the forecast
    date = str(t_ref)[0:10]
    if dt < 10:
        forecast_hour = '0' + str(dt)
    elif dt > 24:
        forecast_hour = '24'
    else:
        forecast_hour = str(dt)
    file = 'Pconvection_' + date + '-' + hour + '_' + forecast_hour + '.00.npz'
    return file
```

generate_dataframe.py

This script retrieves the function previously defined to generate the data frame, first creating a dictionary and then transforming into a *Pandas* dataframe. The function logic_flight is used to filter flights randomly.

The ALL_FT+ files are read and the filtered flights are stored in a list of elements of the class Flight at the same time that the dictionary is created. The list is not required in the generation of the data frame but it was useful in the preliminary implementation of the code, helping to identify the issues encountered.

```
# -*- coding: utf-8 -*-
"""
Created on Thu May 24 18:51:59 2018

@author: Juan Rodriguez Carrillo

TFG: Trajectory predictability analysis under convective weather conditions

generate_dataframe.py takes as input ALL_FT+ files of 2017 and reads them.
Using the functions defined in ALLFT2FLIGHT manipulate the files to extract the
relevant information and stores it in a pandas dataframe.
"""

import os
import datetime
import numpy as np
import pandas as pd
import time


def clear_all():
    """Clears all the variables from the workspace of the spyder application."""
    gl = globals().copy()
    for var in gl:
        if var[0] == '_': continue
        if 'func' in str(globals()[var]): continue
        if 'module' in str(globals()[var]): continue

        del globals()[var]

clear_all()

from ALLFT2FLIGHT import Wpt, Flight, File_manager


def logic_flight(N):
    """
    This function is used to filter random flights.
    The probability of returning TRUE is 1/(N-1).
    If the input argument is 1, 0 or negative, the function returns TRUE
    """
    if N > 0:
        a = np.random.randint(N)
        if a == 0:
            return True
        else:
            return False
    else:
        return True


"""
Data input
```

```
"""

start_time = time.time() #Start computation time


current_date = datetime.date(2017,1,1) #First day of the year
add_day = datetime.timedelta(days = 1) #Add one day


script_dir = os.getcwd()

pathCatFile = os.path.join(script_dir,'ac_category.txt')

ac_dict = {'ac_type': [],
           'ac_cat': []
          } #Generate dictionary to store aircraft categories


f1 = open(pathCatFile,'r') #Loads file to get aircraft category

for line in f1:
    fields = line.split(":")
    ac_type = fields[2]
    ac_cat = fields[6]
    ac_dict['ac_type'].append(ac_type)
    ac_dict['ac_cat'].append(ac_cat)


flights = [] #Initialize empty list of flights
year_flag = True #Flag to stop loop when all days of 2017 have been considered

file_manager = File_manager()

flight_dict ={'prisk':[],
              'delta distance':[],
              'delta time':[],
              'delay in OB':[],
              'delay in TO':[],
              'delay in LD':[],
              'AC cat':[],
              } #Dictionary to generate the dataframe

"""
Read file
"""


while year_flag: #While still year 2017
    current_date_str = str(current_date)
    file_allft = current_date_str[0:4]+current_date_str[5:7]+\
    current_date_str[8:10]+".ALL_FT+"
    pathNestFile = os.path.join(script_dir,'ALL_FT',file_allft)
    if os.path.isfile(pathNestFile): #If the file is available
        f = open(pathNestFile, "r"); #Open the file
        line = f.readline()
        line = f.readline()

        for line in f:

            fields = line.split(";")

            departure_aerodrome = fields[0]
            arrival_aerodrome = fields[1]
            aircraft_ID = fields[2]
            aircraft_operator = fields[3]
            aircraft_type = fields[4]
            aobt = fields[5]
```

```
            eobt = fields[17]
            tail_number = fields[57]
            ftfm_pp = fields[85]  #Filed Tactical Flight Model, Point Profile
            ctfm_pp = fields[113] #Current Tactical Flight Model, Point Profile

            if aircraft_type in ac_dict['ac_type']:
                aircraft_cat = \
                ac_dict['ac_cat'][ac_dict['ac_type'].index(aircraft_type)][0]
            else:
                aicraft_cat = 'M'

            if aircraft_operator == 'BAW' or (aircraft_cat=='L' and logic_flight(5)):

                filed_waypoints = []
                current_waypoints = []

                if len(ctfm_pp) > 0:

                    filed_pp = ftfm_pp.split(' ')
                    for filed_wpt in filed_pp:

                        filed_waypoints.append(Wpt.create_from_string(filed_wpt))


                    current_pp = ctfm_pp.split(' ')
                    for current_wpt in current_pp:

                        current_waypoints.append(Wpt.create_from_string(current_wpt))

                    flight = Flight(departure_aerodrome,arrival_aerodrome,\
                                    aircraft_type,aircraft_cat,eobt,aobt,\
                                    filed_waypoints,\
                                    current_waypoints) #Create flight object
                    flight.compute_convection(file_manager,0.25) #Compute prisk
                    flights.append(flight) #Add flight to the list

                    """
                    Add flight to the dictionary
                    """
                    flight_dict['prisk'].append(flight.prisk) #float
                    if flight.filed_distance > 0:
                        flight_dict['delta distance'].append((\
                        flight.current_distance-flight.filed_distance)\
                        *100/flight.filed_distance) #float
                    else:
                        flight_dict['delta distance'].append(0)

                    estimated_route_time = flight.elt - flight.etot
                    estimated_time = \
                    float(estimated_route_time.seconds)/3600 #float (hours)
                    actual_route_time = flight.alt - flight.atot
                    actual_time = \
                    float(actual_route_time.seconds)/3600 #float (hours)
                    if estimated_time > 0:
                        flight_dict['delta time'].append((actual_time-\
                                    estimated_time)*100/estimated_time) #float
                    else:
                        flight_dict['delta time'].append(0)

                    delayOB = flight.aobt - flight.eobt
                    flight_dict['delay in OB'].append(float(delayOB.days)*24*60\
                              +float(delayOB.seconds)/60) #float (minutes)
                    delayTO = flight.atot - flight.etot
                    flight_dict['delay in TO'].append(float(delayTO.days)*24*60\
                              +float(delayTO.seconds)/60) #float (minutes)
                    delayLD = flight.alt - flight.elt
                    flight_dict['delay in LD'].append(float(delayTO.days)*24*60\
```

```
                                +float(delayLD.seconds)/60) #float (minutes)

                    flight_dict['AC cat'].append(flight.AC_cat) #string
        f.close() #close ALL_FT+ file
    current_date += add_day #Next day
    if current_date.year == 2018:
        year_flag = False #End loop when 2017 is finished


DF = pd.DataFrame(data=flight_dict) #Generate dataframe from dictionary

pathPrevDF = os.path.join(script_dir,'dataframe')

if os.path.isfile(pathPrevDF): #If dataframe already exists
    PrevDF = pd.read_pickle('dataframe') #Loads previous dataframe
    DF = pd.concat([PrevDF,DF]) #Concatenate dataframes


DF.to_pickle('dataframe') #Store dataframe


print("--- %s seconds ---" % (time.time() - start_time)) #Display computation time
```

correlate_prisk.py

The final script takes the generated data frame and obtains the correlations. It also displays histograms of the variables analyzed.

```
# -*- coding: utf-8 -*-
"""
Created on Fri May 25 11:35:43 2018

@author: Juan Rodriguez Carrillo

TFG: Trajectory predictability analysis under convective weather conditions

correlate_prisk.py takes as input the pandas dataframe obtained in generate_dataframe.py.
It uses linear regressions to correlate the variables of interest and generates plots
to show the correlations. It also generates histograms of the variables studied.
"""

import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
from sklearn import linear_model
import scipy.stats



def linear_regression(x, y, prob):
    """
    Return the linear regression parameters and their <prob> confidence intervals.
    """
    x = np.array(x)
    y = np.array(y)
    n = len(x)
    xy = x * y
    xx = x * x

    # estimates

    b1 = (xy.mean() - x.mean() * y.mean()) / (xx.mean() - x.mean()**2)
    b0 = y.mean() - b1 * x.mean()
    s2 = 1./n * sum([(y[i] - b0 - b1 * x[i])**2 for i in range(n)])

    #confidence intervals
    alpha = 1 - prob
    c = -1 * scipy.stats.t.ppf(alpha/2.,n-2)
    bb1 = c * (s2 / ((n-2) * (xx.mean() - (x.mean())**2)))**.5
    bb0 = c * ((s2 / (n-2)) * (1 + (x.mean())**2 /\
                (xx.mean() - (x.mean())**2)))**.5

    print('b0 = %f +- %f'%(b0,bb0))
    print('b1 = %f +- %f'%(b1,bb1))
    return None

def clear_all():
    """Clears all the variables from the workspace of the spyder application."""
    gl = globals().copy()
    for var in gl:
        if var[0] == '_': continue
        if 'func' in str(globals()[var]): continue
        if 'module' in str(globals()[var]): continue

        del globals()[var]

clear_all()


```

```
DF = pd.read_pickle('dataframe') #Get the generated dataframe

DF_L = DF.loc[DF['AC cat'] == 'L'] #Category L (Light)
DF_M = DF.loc[DF['AC cat'] == 'M'] #Category M (Medium)
DF_H = DF.loc[DF['AC cat'] == 'H'] #Category H (Heavy)

""" Regressions """
#Category M

regr1 = linear_model.LinearRegression()
regr1.fit(DF_M['prisk'].values.reshape(-1,1),DF_M['delay in OB'].values.reshape(-1,1))
x1 = np.array(np.linspace(0,DF_M['prisk'].max(),400))
y1 = regr1.predict(x1.reshape(-1,1))


plt.scatter(DF_M['prisk'], DF_M['delay in OB'],  color='black', label='Flight data')
plt.plot(x1,y1.tolist(), color='red', label='Linear regression', linewidth=3)
plt.grid()
plt.ylim(-50,200)
plt.xlabel('$p_{risk} (km)$')
plt.ylabel('Delay in Off-Block (min)')
plt.title('Category M')
plt.legend(loc = 1)
plt.savefig('prisk_delayOB_M.png')
plt.show()

regr2 = linear_model.LinearRegression()
regr2.fit(DF_M['prisk'].values.reshape(-1,1),DF_M['delay in TO'].values.reshape(-1,1))
x2 = np.array(np.linspace(0,DF_M['prisk'].max(),400))
y2 = regr2.predict(x2.reshape(-1,1))

plt.scatter(DF_M['prisk'], DF_M['delay in TO'],  color='black', label='Flight data')
plt.plot(x2,y2.tolist(), color='red', label='Linear regression', linewidth=3)
plt.grid()
plt.ylim(-50,200)
plt.xlabel('$p_{risk} (km)$')
plt.ylabel('Delay in Take-Off (min)')
plt.title('Category M')
plt.legend(loc = 1)
plt.savefig('prisk_delayTO_M.png')
plt.show()

regr3 = linear_model.LinearRegression()
regr3.fit(DF_M['prisk'].values.reshape(-1,1),DF_M['delay in LD'].values.reshape(-1,1))
x3 = np.array(np.linspace(0,DF_M['prisk'].max(),400))
y3 = regr3.predict(x3.reshape(-1,1))

plt.scatter(DF_M['prisk'], DF_M['delay in LD'],  color='black', label='Flight data')
plt.plot(x3,y3.tolist(), color='red', label='Linear regression', linewidth=3)
plt.grid()
plt.ylim(-100,250)
plt.xlabel('$p_{risk} (km)$')
plt.ylabel('Delay in Landing (min)')
plt.title('Category M')
plt.legend(loc = 1)
plt.savefig('prisk_delayLD_M.png')
plt.show()

regr4 = linear_model.LinearRegression()
regr4.fit(DF_M['prisk'].values.reshape(-1,1),DF_M['delta distance'].values.reshape(-1,1))
x4 = np.array(np.linspace(0,DF_M['prisk'].max(),400))
y4 = regr4.predict(x4.reshape(-1,1))

plt.scatter(DF_M['prisk'], DF_M['delta distance'],  color='black', label='Flight data')
plt.plot(x4,y4.tolist(), color='red', label='Linear regression', linewidth=3)
plt.grid()
plt.ylim(-100,150)
```

```
plt.xlabel('$p_{risk} (km)$')
plt.ylabel('$\Delta L (\%)$')
plt.title('Category M')
plt.legend(loc = 1)
plt.savefig('prisk_deltadistance_M.png')
plt.show()

regr5 = linear_model.LinearRegression()
regr5.fit(DF_M['prisk'].values.reshape(-1,1),DF_M['delta time'].values.reshape(-1,1))
x5 = np.array(np.linspace(0,DF_M['prisk'].max(),400))
y5 = regr5.predict(x5.reshape(-1,1))

plt.scatter(DF_M['prisk'], DF_M['delta time'],  color='black', label='Flight data')
plt.plot(x5,y5.tolist(), color='red', label='Linear regression', linewidth=3)
plt.grid()
plt.ylim(-100,150)
plt.xlabel('$p_{risk} (km)$')
plt.ylabel('$\Delta t (\%)$')
plt.title('Category M')
plt.legend(loc = 1)
plt.savefig('prisk_deltatime_M.png')
plt.show()

#Category H

regr6 = linear_model.LinearRegression()
regr6.fit(DF_H['prisk'].values.reshape(-1,1),DF_H['delay in OB'].values.reshape(-1,1))
x6 = np.array(np.linspace(0,DF_H['prisk'].max(),400))
y6 = regr6.predict(x6.reshape(-1,1))

plt.scatter(DF_H['prisk'], DF_H['delay in OB'],  color='black', label='Flight data')
plt.plot(x6,y6.tolist(), color='red', label='Linear regression', linewidth=3)
plt.grid()
plt.ylim(-50,300)
plt.xlabel('$p_{risk} (km)$')
plt.ylabel('Delay in Off-Block (min)')
plt.title('Category H')
plt.legend(loc = 1)
plt.savefig('prisk_delayOB_H.png')
plt.show()

regr7 = linear_model.LinearRegression()
regr7.fit(DF_H['prisk'].values.reshape(-1,1),DF_H['delay in TO'].values.reshape(-1,1))
x7 = np.array(np.linspace(0,DF_H['prisk'].max(),400))
y7 = regr7.predict(x7.reshape(-1,1))

plt.scatter(DF_H['prisk'], DF_H['delay in TO'],  color='black', label='Flight data')
plt.plot(x7,y7.tolist(), color='red', label='Linear regression', linewidth=3)
plt.grid()
plt.ylim(-50,300)
plt.xlabel('$p_{risk} (km)$')
plt.ylabel('Delay in Take-Off (min)')
plt.title('Category H')
plt.legend(loc = 1)
plt.savefig('prisk_delayTO_H.png')
plt.show()

regr8 = linear_model.LinearRegression()
regr8.fit(DF_H['prisk'].values.reshape(-1,1),DF_H['delay in LD'].values.reshape(-1,1))
x8 = np.array(np.linspace(0,DF_H['prisk'].max(),400))
y8 = regr8.predict(x8.reshape(-1,1))

plt.scatter(DF_H['prisk'], DF_H['delay in LD'],  color='black', label='Flight data')
plt.plot(x8,y8.tolist(), color='red', label='Linear regression', linewidth=3)
plt.grid()
plt.ylim(-200,500)
plt.xlabel('$p_{risk} (km)$')
```

```
plt.ylabel('Delay in Landing (min)')
plt.title('Category H')
plt.legend(loc = 1)
plt.savefig('prisk_delayLD_H.png')
plt.show()

regr9 = linear_model.LinearRegression()
regr9.fit(DF_H['prisk'].values.reshape(-1,1),DF_H['delta distance'].values.reshape(-1,1))
x9 = np.array(np.linspace(0,DF_H['prisk'].max(),400))
y9 = regr9.predict(x9.reshape(-1,1))

plt.scatter(DF_H['prisk'], DF_H['delta distance'],  color='black', label='Flight data')
plt.plot(x9,y9.tolist(), color='red', label='Linear regression', linewidth=3)
plt.grid()
plt.ylim(-50,50)
plt.xlabel('$p_{risk} (km)$')
plt.ylabel('$\Delta L (\%)$')
plt.title('Category H')
plt.legend(loc = 1)
plt.savefig('prisk_deltadistance_H.png')
plt.show()

regr10 = linear_model.LinearRegression()
regr10.fit(DF_H['prisk'].values.reshape(-1,1),DF_H['delta time'].values.reshape(-1,1))
x10 = np.array(np.linspace(0,DF_H['prisk'].max(),400))
y10 = regr10.predict(x10.reshape(-1,1))

plt.scatter(DF_H['prisk'], DF_H['delta time'],  color='black', label='Flight data')
plt.plot(x10,y10.tolist(), color='red', label='Linear regression', linewidth=3)
plt.ylim(-20,80)
plt.grid()
plt.xlabel('$p_{risk} (km)$')
plt.ylabel('$\Delta t (\%)$')
plt.title('Category H')
plt.legend(loc = 1)
plt.savefig('prisk_deltatime_H.png')
plt.show()

#Category L

regr11 = linear_model.LinearRegression()
regr11.fit(DF_L['prisk'].values.reshape(-1,1),DF_L['delay in OB'].values.reshape(-1,1))
x11 = np.array(np.linspace(0,DF_L['prisk'].max(),400))
y11 = regr11.predict(x11.reshape(-1,1))

plt.scatter(DF_L['prisk'], DF_L['delay in OB'],  color='black', label='Flight data')
plt.plot(x11,y11.tolist(), color='red', label='Linear regression', linewidth=3)
plt.grid()
plt.ylim(-100,100)
plt.xlabel('$p_{risk} (km)$')
plt.ylabel('Delay in Off-Block (min)')
plt.title('Category L')
plt.legend(loc = 1)
plt.savefig('prisk_delayOB_L.png')
plt.show()

regr12 = linear_model.LinearRegression()
regr12.fit(DF_L['prisk'].values.reshape(-1,1),DF_L['delay in TO'].values.reshape(-1,1))
x12 = np.array(np.linspace(0,DF_L['prisk'].max(),400))
y12 = regr12.predict(x12.reshape(-1,1))

plt.scatter(DF_L['prisk'], DF_L['delay in TO'],  color='black', label='Flight data')
plt.plot(x12,y12.tolist(), color='red', label='Linear regression', linewidth=3)
plt.grid()
plt.ylim(-100,100)
plt.xlabel('$p_{risk} (km)$')
plt.ylabel('Delay in Take-Off (min)')
```

```
plt.title('Category L')
plt.legend(loc = 1)
plt.savefig('prisk_delayTO_L.png')
plt.show()

regr13 = linear_model.LinearRegression()
regr13.fit(DF_L['prisk'].values.reshape(-1,1),DF_L['delay in LD'].values.reshape(-1,1))
x13 = np.array(np.linspace(0,DF_L['prisk'].max(),400))
y13 = regr13.predict(x13.reshape(-1,1))

plt.scatter(DF_L['prisk'], DF_L['delay in LD'],  color='black', label='Flight data')
plt.plot(x13,y13.tolist(), color='red', label='Linear regression', linewidth=3)
plt.grid()
plt.ylim(-200,200)
plt.xlabel('$p_{risk} (km)$')
plt.ylabel('Delay in Landing (min)')
plt.title('Category L')
plt.legend(loc = 1)
plt.savefig('prisk_delayLD_L.png')
plt.show()

regr14 = linear_model.LinearRegression()
regr14.fit(DF_L['prisk'].values.reshape(-1,1),DF_L['delta distance'].values.reshape(-1,1))
x14 = np.array(np.linspace(0,DF_L['prisk'].max(),400))
y14 = regr14.predict(x14.reshape(-1,1))

plt.scatter(DF_L['prisk'], DF_L['delta distance'],  color='black', label='Flight data')
plt.plot(x14,y14.tolist(), color='red', label='Linear regression', linewidth=3)
plt.grid()
plt.ylim(-100,150)
plt.xlabel('$p_{risk} (km)$')
plt.ylabel('$\Delta L (\%)$')
plt.title('Category L')
plt.legend(loc = 1)
plt.savefig('prisk_deltadistance_L.png')
plt.show()

regr15 = linear_model.LinearRegression()
regr15.fit(DF_L['prisk'].values.reshape(-1,1),DF_L['delta time'].values.reshape(-1,1))
x15 = np.array(np.linspace(0,DF_L['prisk'].max(),400))
y15 = regr15.predict(x15.reshape(-1,1))

plt.scatter(DF_L['prisk'], DF_L['delta time'],  color='black', label='Flight data')
plt.plot(x15,y15.tolist(), color='red', label='Linear regression', linewidth=3)
plt.grid()
plt.ylim(-100,150)
plt.xlabel('$p_{risk} (km)$')
plt.ylabel('$\Delta t (\%)$')
plt.title('Category L')
plt.legend(loc = 1)
plt.savefig('prisk_deltatime_L.png')
plt.show()




""" Histograms """
#Category M

plt.hist(DF_M['prisk'])
plt.xlabel('$p_{risk} (km)$')
plt.ylabel('Number of flights')
plt.title('Category M')
plt.savefig('hist_prisk_M.png')
plt.show()

```

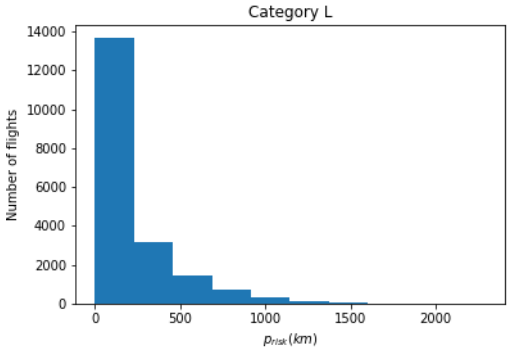
```
plt.hist(DF_M['delay in OB'], range=(-50,150))
plt.xlabel('Delay in Off-Block (min)')
plt.ylabel('Number of flights')
plt.title('Category M')
plt.savefig('hist_delayOB_M.png')
plt.show()

plt.hist(DF_M['delay in TO'], range=(-50,150))
plt.xlabel('Delay in Take-Off (min)')
plt.ylabel('Number of flights')
plt.title('Category M')
plt.savefig('hist_delayTO_M.png')
plt.show()

plt.hist(DF_M['delay in LD'], range=(-50,150))
plt.xlabel('Delay in Landing (min)')
plt.ylabel('Number of flights')
plt.title('Category M')
plt.savefig('hist_delayLD_M.png')
plt.show()

plt.hist(DF_M['delta distance'], range = (-30,40))
plt.xlabel('$\Delta L (\%)$')
plt.ylabel('Number of flights')
plt.title('Category M')
plt.savefig('hist_deltadistance_M.png')
plt.show()

plt.hist(DF_M['delta time'], range(-50,60))
plt.xlabel('$\Delta t (\%)$')
plt.ylabel('Number of flights')
plt.title('Category M')
plt.savefig('hist_deltatime_M.png')
plt.show()

#Category H

plt.hist(DF_H['prisk'])
plt.xlabel('$p_{risk} (km)$')
plt.ylabel('Number of flights')
plt.title('Category H')
plt.savefig('hist_prisk_H.png')
plt.show()

plt.hist(DF_H['delay in OB'], range=(-50,150))
plt.xlabel('Delay in Off-Block (min)')
plt.ylabel('Number of flights')
plt.title('Category H')
plt.savefig('hist_delayOB_H.png')
plt.show()

plt.hist(DF_H['delay in TO'], range=(-50,150))
plt.xlabel('Delay in Take-Off (min)')
plt.ylabel('Number of flights')
plt.title('Category H')
plt.savefig('hist_delayTO_H.png')
plt.show()

plt.hist(DF_H['delay in LD'], range=(-50,150))
plt.xlabel('Delay in Landing (min)')
plt.ylabel('Number of flights')
plt.title('Category H')
plt.savefig('hist_delayLD_H.png')
plt.show()

plt.hist(DF_H['delta distance'], range = (-20,20))
plt.xlabel('$\Delta L (\%)$')
```

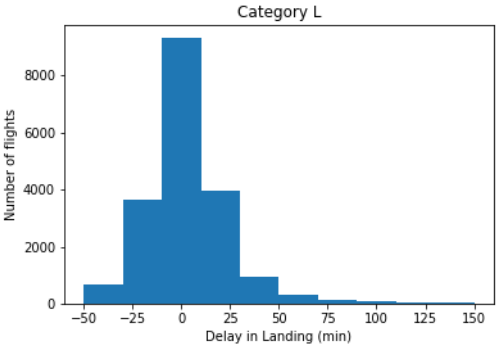
```
plt.ylabel('Number of flights')
plt.title('Category H')
plt.savefig('hist_deltadistance_H.png')
plt.show()

plt.hist(DF_H['delta time'], range = (-30,20))
plt.xlabel('$\Delta t (\%)$')
plt.ylabel('Number of flights')
plt.title('Category H')
plt.savefig('hist_deltatime_H.png')
plt.show()

#Category L

plt.hist(DF_L['prisk'])
plt.xlabel('$p_{risk} (km)$')
plt.ylabel('Number of flights')
plt.title('Category L')
plt.savefig('hist_prisk_L.png')
plt.show()

plt.hist(DF_L['delay in OB'], range=(-80,100))
plt.xlabel('Delay in Off-Block (min)')
plt.ylabel('Number of flights')
plt.title('Category L')
plt.savefig('hist_delayOB_L.png')
plt.show()

plt.hist(DF_L['delay in TO'], range=(-80,100))
plt.xlabel('Delay in Take-Off (min)')
plt.ylabel('Number of flights')
plt.title('Category L')
plt.savefig('hist_delayTO_L.png')
plt.show()

plt.hist(DF_L['delay in LD'], range=(-50,150))
plt.xlabel('Delay in Landing (min)')
plt.ylabel('Number of flights')
plt.title('Category L')
plt.savefig('hist_delayLD_L.png')
plt.show()

plt.hist(DF_L['delta distance'], range(-50,100))
plt.xlabel('$\Delta L (\%)$')
plt.ylabel('Number of flights')
plt.title('Category L')
plt.savefig('hist_deltadistance_L.png')
plt.show()

plt.hist(DF_L['delta time'], range(-50,100))
plt.xlabel('$\Delta t (\%)$')
plt.ylabel('Number of flights')
plt.title('Category L')
plt.savefig('hist_deltatime_L.png')
plt.show()


"""
Variance
"""
print('-------- VARIANCE --------')
print('Category L')
print(DF_L.var())
print('Category M')
print(DF_M.var())
print('Category H')
print(DF_H.var())
```

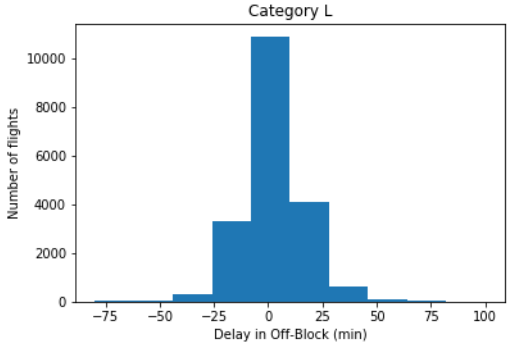
```
464
465
466 """
467 Standard deviation
468 """
469 print('------ STD DEVIATION -----')
470 print('Category L')
471 print(DF_L.std())
472 print('Category M')
473 print(DF_M.std())
474 print('Category H')
475 print(DF_H.std())
476
477 """
478 Linear regression parameters
479 """
480 print('----- LR PARAMETERS -----')
481 print('$$$ Category L $$$')
482 print('Delay in OB')
483 linear_regression(DF_L['prisk'],DF_L['delay in OB'],0.95)
484 print('Delay in TO')
485 linear_regression(DF_L['prisk'],DF_L['delay in TO'],0.95)
486 print('Delay in LD')
487 linear_regression(DF_L['prisk'],DF_L['delay in LD'],0.95)
488 print('Delta distance')
489 linear_regression(DF_L['prisk'],DF_L['delta distance'],0.95)
490 print('Delta time')
491 linear_regression(DF_L['prisk'],DF_L['delta time'],0.95)
492
493 print('$$$ Category M $$$')
494 print('Delay in OB')
495 linear_regression(DF_M['prisk'],DF_M['delay in OB'],0.95)
496 print('Delay in TO')
497 linear_regression(DF_M['prisk'],DF_M['delay in TO'],0.95)
498 print('Delay in LD')
499 linear_regression(DF_M['prisk'],DF_M['delay in LD'],0.95)
500 print('Delta distance')
501 linear_regression(DF_M['prisk'],DF_M['delta distance'],0.95)
502 print('Delta time')
503 linear_regression(DF_M['prisk'],DF_M['delta time'],0.95)
504
505 print('$$$ Category H $$$')
506 print('Delay in OB')
507 linear_regression(DF_H['prisk'],DF_H['delay in OB'],0.95)
508 print('Delay in TO')
509 linear_regression(DF_H['prisk'],DF_H['delay in TO'],0.95)
510 print('Delay in LD')
511 linear_regression(DF_H['prisk'],DF_H['delay in LD'],0.95)
512 print('Delta distance')
513 linear_regression(DF_H['prisk'],DF_H['delta distance'],0.95)
514 print('Delta time')
515 linear_regression(DF_H['prisk'],DF_H['delta time'],0.95)
```

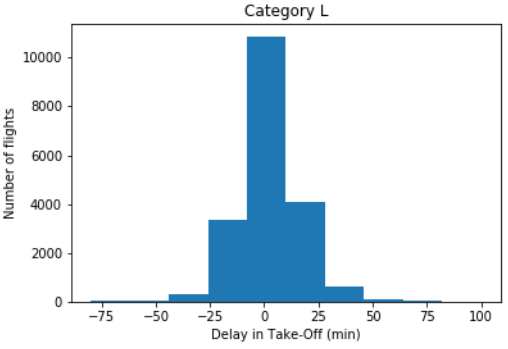
Appendix C - Histograms

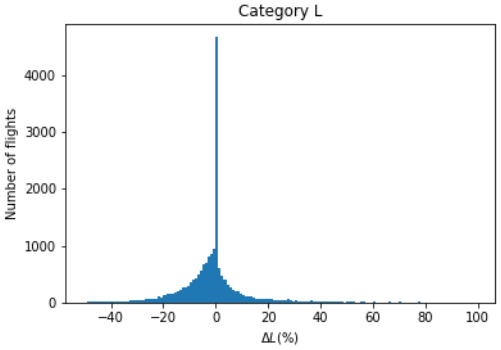
Category L

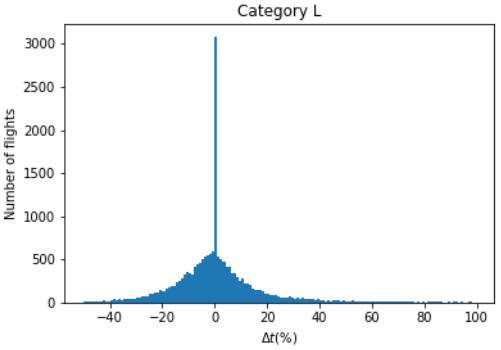




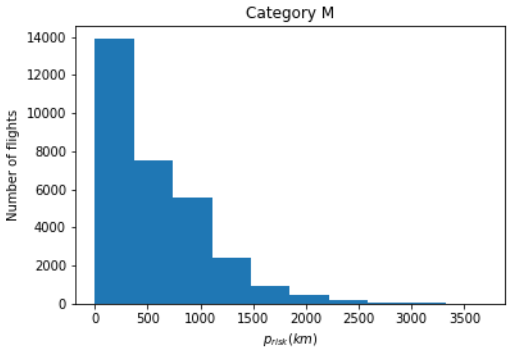


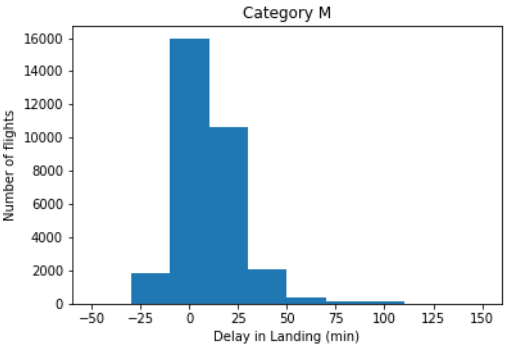


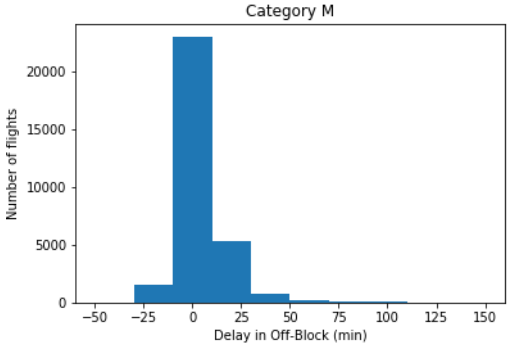


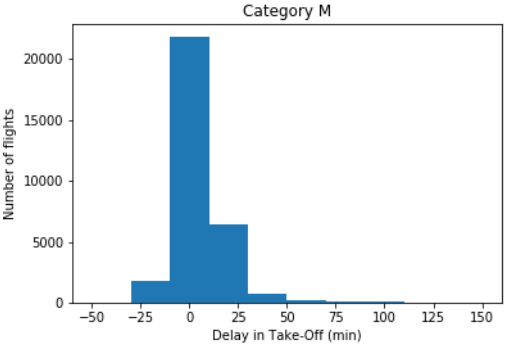


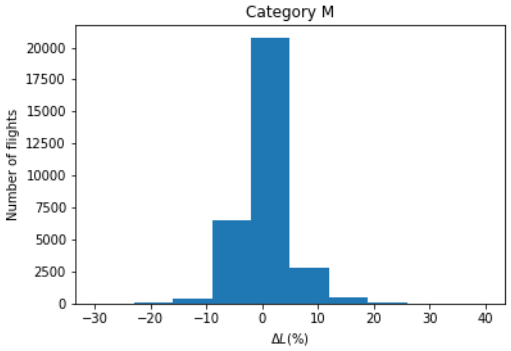
Category M

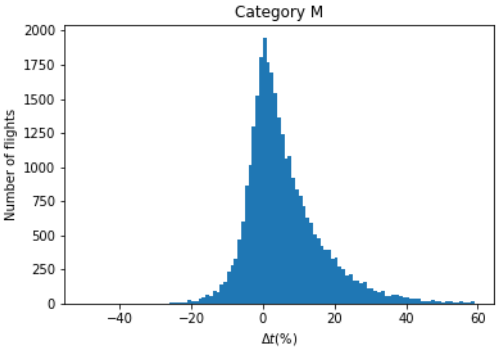

Category M
Number of flights
p_{risk} (km)


Category M
Number of flights
Delay in Landing (min)

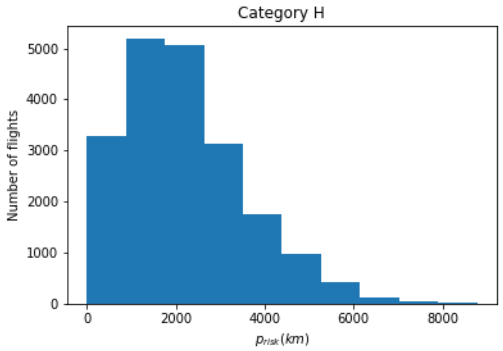

Category M
Number of flights
Delay in Off-Block (min)

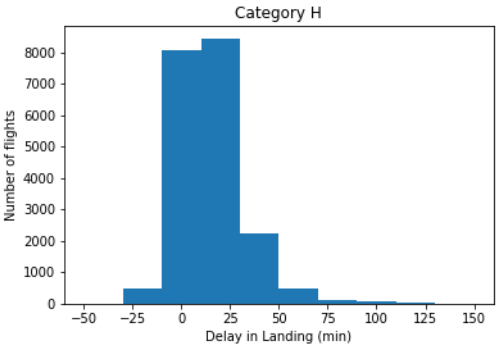

Category M
Number of flights
Delay in Take-Off (min)

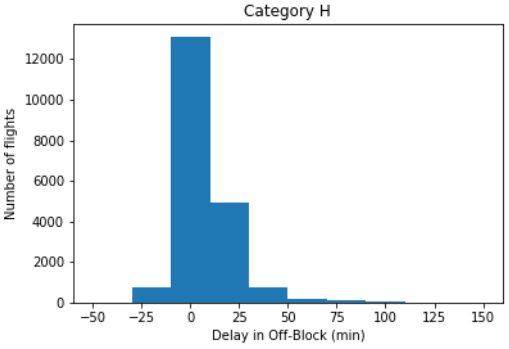

Category M
Number of flights
ΔL (%)

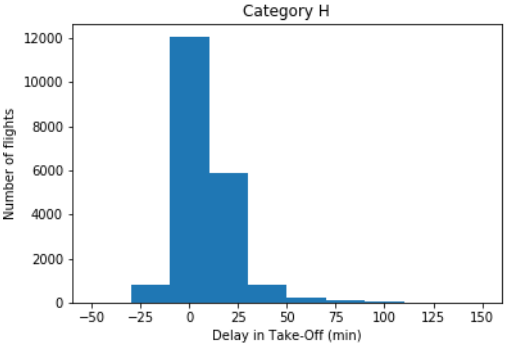

Category M
Number of flights
Δt(%)

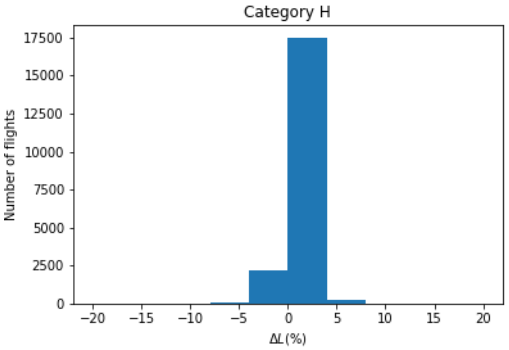
Category H

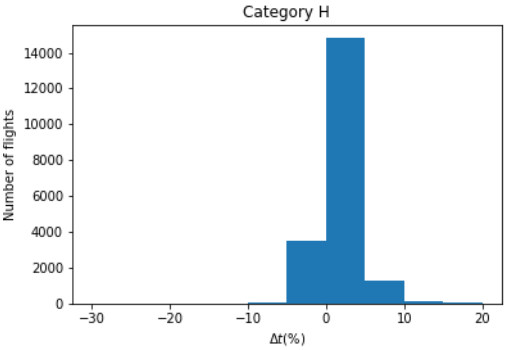

Category H
Number of flights
0
1000
2000
3000
4000
5000
0
2000
4000
6000
8000
p_{risk} (km)


Category H
Number of flights
0
1000
2000
3000
4000
5000
6000
7000
8000
−50
−25
0
25
50
75
100
125
150
Delay in Landing (min)


Category H
Number of flights
0
2000
4000
6000
8000
10000
12000
−50
−25
0
25
50
75
100
125
150
Delay in Off-Block (min)


Category H
Number of flights
0
2000
4000
6000
8000
10000
12000
−50
−25
0
25
50
75
100
125
150
Delay in Take-Off (min)


Category H
Number of flights
0
2500
5000
7500
10000
12500
15000
17500
−20
−15
−10
−5
0
5
10
15
20
ΔL (%)


Category H
Number of flights
0
2000
4000
6000
8000
10000
12000
14000
−30
−20
−10
0
10
20
Δt(%)